Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

TRABALHO DE CONCLUSÃO DE CURSO

ESTIMATIVA DE FLUXO DE TRÂNSITO UTILIZANDO PROCESSAMENTO DE VÍDEO

Abmael Vilar Barros

Campina Grande, Paraíba

Março de 2022

ABMAEL VILAR BARROS

ESTIMATIVA DE FLUXO DE TRÂNSITO UTILIZANDO PROCESSAMENTO DE VÍDEO

Trabalho de Conclusão de Curso submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduado em Engenharia Elétrica.

Área de Concentração: Processamento de Vídeo

Orientadora: Prof. Luciana Ribeiro Veloso, Dra.

Campina Grande, Paraíba

Março de 2022

ABMAEL VILAR BARROS

ESTIMATIVA DE FLUXO DE TRÂNSITO UTILIZANDO PROCESSAMENTO DE VÍDEO

Trabalho de Conclusão de Curso submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduado em Engenharia Elétrica.

Trabalho aprovado em: _____/_____/_____

_____

Orientadora: Prof. Luciana Ribeiro Veloso,
Dra.

_____

Convidado: Prof. Edmar Candeia Gurjão,
Dr.

Campina Grande, Paraíba

Março de 2022

Dedico este trabalho a meus pais e todos que acreditaram no meu potencial.

Agradecimentos

Agradeço à minha orientadora Professora Luciana Ribeiro pela orientação e apoio.

Agradeço a Léo pelo apoio com a parte técnica utilizada em algumas partes do trabalho.

“O ontem é história. O amanhã é um mistério, mas o hoje é uma dádiva, é por isso que se chama presente.”

- Hugo, Kung Fu Panda.

Resumo

A concepção de cidades inteligentes, partem do princípio de que os sistemas de infraestrutura públicos como, energia, água, rodovias dentre outros tipos, podem ser melhores se projetadas, executados e administrados a partir de informações coletadas com o uso de tecnologia integrada. Dentre as tecnologias que podem ser implementadas, o uso de câmeras de vigilância já é uma realidade atualmente. Em muitas cidades do Brasil e do mundo, a avaliação do fluxo de veículos em vias é realizada através da análise de câmeras de vídeo. Sendo assim, é possível utilizar a tecnologia de inteligência artificial para automatizar essa análise e, assim, auxiliar os gestores na tomada de decisão de políticas públicas. Neste trabalho é apresentado uma forma automatizada de análise de fluxo de veículos, utilizando uma rede neural convolucional YOLO para a analisar a quantidade de veículos que trafega por uma via específica, fazendo a contagem automática de cada tipo de veículo em um determinado período.

Palavras-chave: Processamento de Imagens, Redes Neurais Convolucionais, Python, Inteligência Artificial, *YOLO, OpenCV*.

ABSTRACT

The design of smart cities assumes that public infrastructure systems such as energy, water, highways, among other types, can be better designed, executed and managed from information collected using integrated technology. Among the technologies that can be implemented, the use of surveillance cameras is already a reality today. In many cities in Brazil and around the world, the evaluation of the flow of vehicles on roads is carried out through the analysis of video cameras. Therefore, it is possible to use artificial intelligence technology to automate this analysis and thus assist managers in public policy decision-making. In this work, an automated form of vehicle flow analysis is presented, using a YOLO convolutional neural network to analyze the number of vehicles traveling through a specific road, automatically counting each type of vehicle in a given period.

Keywords: Image Processing, Convolutional Neural Networks, Python, Artificial Intelligence, *YOLO, OpenCV*.

LISTA DE ABREVIATURAS E SIGLAS

TCC	Trabalho de Conclusão de Curso
YOLO	*You Only Look Once*
ReLU	*Rectified Linear Unity*
CNN	*Convolutional Neural Networks*
GPU	*Graphic Processor Unity*
OPENCV	*Open Source Computer Vision*
CUDA	*Compute Unified Device Architecture*
CUDNN	*CUDA Deep Neural Network*
FPS	*Frame Per Second*
BLOB	*Binary Large Object*
CSPDarknet	*convolutional neural network and backbone for object detection*
SSPNet	*Scale Selection Pyramid Network*
PANet	*Path Aggregation Network*
CIOU	*Complete Intersection Over Union*

LISTA DE ILUSTRAÇÕES

LISTA DE EQUAÇÕES

LISTA DE TABELAS

Sumário

1 INTRODUÇÃO

A mobilidade urbana é um conceito bastante discutido nas políticas públicas que envolvem o planejamento das cidades. Ela é definida como qualquer tipo de movimento – a pé, de carro, ônibus, bicicleta, skate, cadeira de rodas, trem ou metrô – que tenha como finalidade o deslocamento de um ponto a outro em um espaço geográfico. Diversos países têm estado atentos à problemática da mobilidade urbana e os custos relacionados. O relatório preparado pelo Conselho Econômico Nacional (*National Economic Council*) americano, publicado em 2014, mostrou ser crítico para os EUA investir em uma infraestrutura forte e eficiente, para poder manter a competitividade do país em um mercado global. (C. GORGULHO; M.TREDINNICK, 2020)

No Brasil problemas relacionados à mobilidade urbana geram prejuízos econômicos além de impactos socioambientais. Estudos mostram que mais de 9 milhões de brasileiros demoram mais de uma hora para chegar ao trabalho e a situação se agrava a cada dia. Desta forma, o Brasil perde R$ 267 bilhões por ano com congestionamentos. (C. GORGULHO; M.TREDINNICK, 2020)

Diante deste cenário, o conceito de cidade inteligente visa resolver problemas urgentes que acompanham a urbanização progressiva, como por exemplo, o tráfego excessivo gerando congestionamentos, sobrecarga de sistemas de energia e água, habitação e outros problemas de infraestrutura. (O. GASSMANN; J.BOHM; M.PLAMIÉ, 2019)

Nesse contexto, o monitoramento de trânsito é uma parte fundamental para resolução de problemas de escoamento de veículos em vias e para tomar melhores decisões de infraestrutura de tráfego além de acompanhamento infrações e acidentes entre outras características pertinentes a passagem de veículos automotores e outros meios de transporte em vias públicas.

Existem diversas formas de se realizar este monitoramento, sendo elas manuais ou automáticas. Com o avanço das tecnologias de captação e transmissão de vídeos, as grandes cidades já utilizam sistemas de captura de vídeo com câmeras e um fluxo de *stream* contínuo. (M. MELO, 2021)

A partir destes vídeos gerados quase que em tempo real é possível utilizar algoritmos pré-treinados para realizar tarefas que antes dependeriam de um operador humano, como, por exemplo, contagem de veículos, análise de fluxo e detecção de infrações.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

- Criar um sistema capaz de identificar e contabilizar separadamente veículos em autovias a partir de câmeras de vigilância.

1.1.2 Objetivos Específicos

- Processar vídeos para analisar o fluxo de trânsito e fazer a contagem de veículos.
- Estimar o fluxo de veículos ao longo do dia.

O algoritmo de identificação e contagem de veículos foi desenvolvido em linguagem Python juntamente com a biblioteca *OpenCV*.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está organizado em 5 capítulos. No capítulo 1 é feita uma introdução do trabalho e são apresentados os objetivos. Em seguida, no capítulo 2, é descrito um conjunto de conceitos necessários para a compreensão dos assuntos abordados no trabalho. São descritos conceitos sobre Redes Neurais Artificiais, Redes Neurais Convolucionais, alguns conceitos sobre Processamento Digital de Imagens e Cidades Inteligentes. O capítulo 3 tem-se a descrição da metodologia de implantação da solução descrevendo os passos necessários. Por fim, os capítulos 4 e 5, são feitas as considerações sobre o resultado e a conclusão do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 REDES NEURAIS ARTIFICIAIS

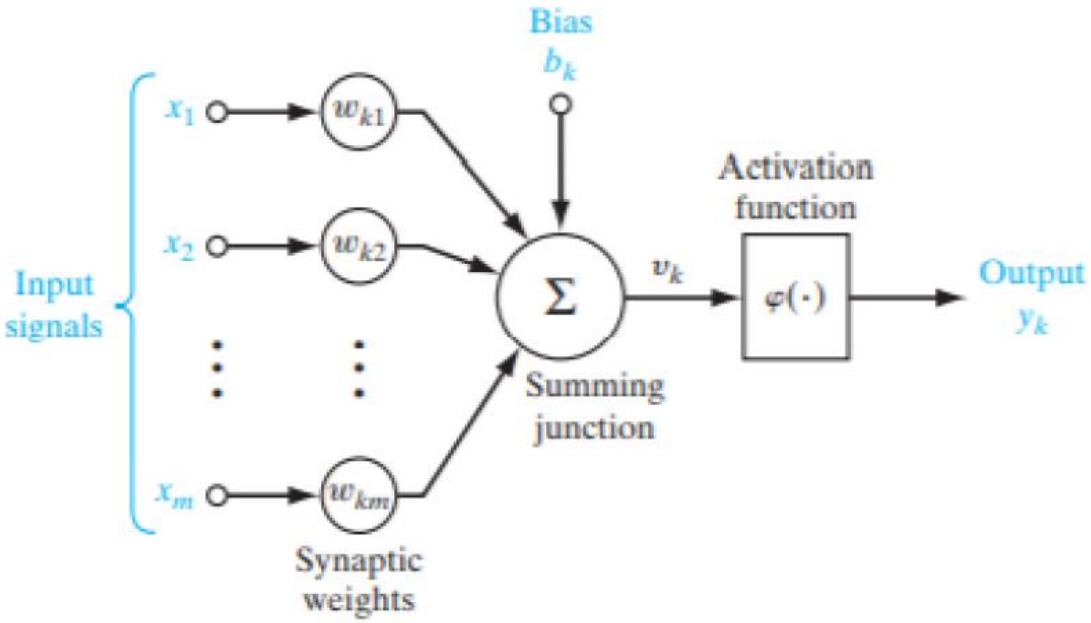
Uma Rede Neural Artificial, segundo (HAYKIN; HAYKIN, 2009), pode ser definida como um processador maciçamente e paralelamente distribuído, constituído de unidades de processamento simples, que têm a propensão natural de armazenar conhecimento experimental e torná-lo disponível para o uso. Ela tenta simular computacionalmente o comportamento do sistema de conexões neurais humano, tentando imitar a capacidade de aprendizagem a partir de estímulos e a capacidade de armazenar o conhecimento adquirido. Redes neurais artificiais são

uma estrutura complexa e interconectada construída a partir de uma estrutura mais simples chamada neurônio artificial.

2.1.1 Neurônio

Um neurônio artificial é a menor unidade da rede que tenta imitar o funcionamento de neurônio biológico, conforme ilustrado na Figura 1.

Figura 1 - Modelo de um Neurônio Artificial.



Fonte: (HAYKIN; HAIKIN, 2009).

O modelo neural é composto de três elementos básicos, segundo (HAYKIN; HAYKIN, 2009):

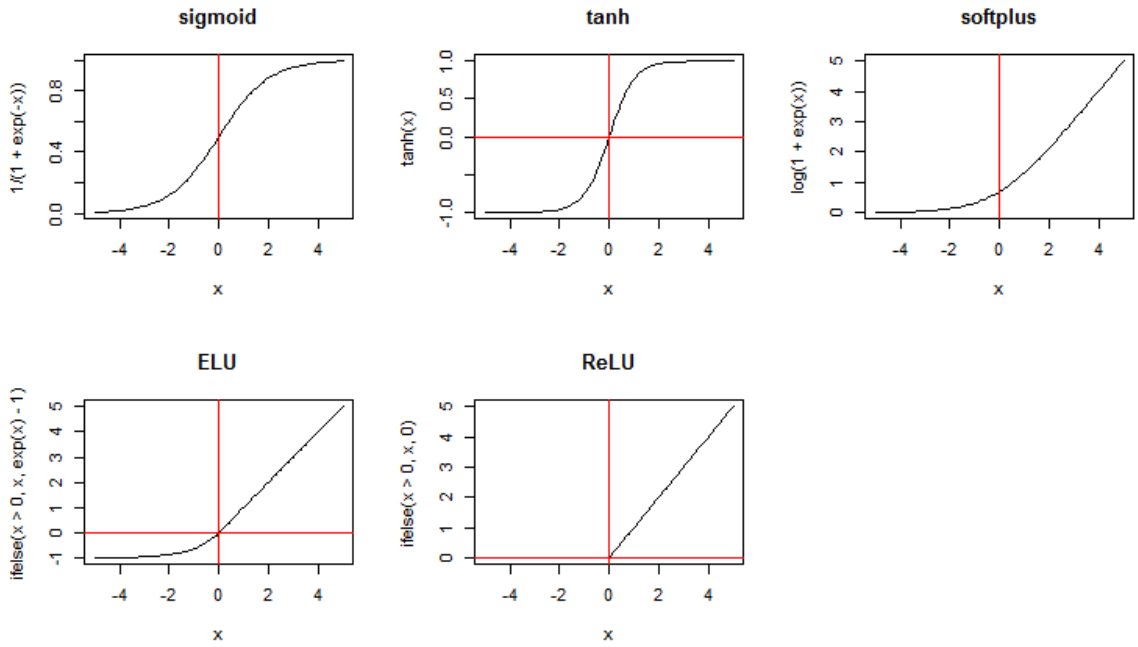
1) Um conjunto de sinapses, representado por entradas x_i.
2) Um somador que realiza a soma ponderada das entradas.
3) Uma função de ativação que computa amplitude de saída do neurônio.

A saída do neurônio é obtida somando-se as entradas previamente ponderadas e aplicando o resultado na função de ativação $\varphi(.)$.

2.1.2 Função de Ativação

A função de ativação representa a resposta aos impulsos estimulados pelas entradas de um neurônio artificial, e normalmente representa uma transformação não linear. Existem diversas funções que podem ser utilizadas como função de ativação. A escolha da função de ativação deve ser realizada com cuidado e sempre observando as características do problema a ser tratado. Exemplos de algumas das principais funções de ativações, que são utilizadas na literatura especializada, são ilustradas na Figura 2.

Figura 2 - Funções de Ativação.



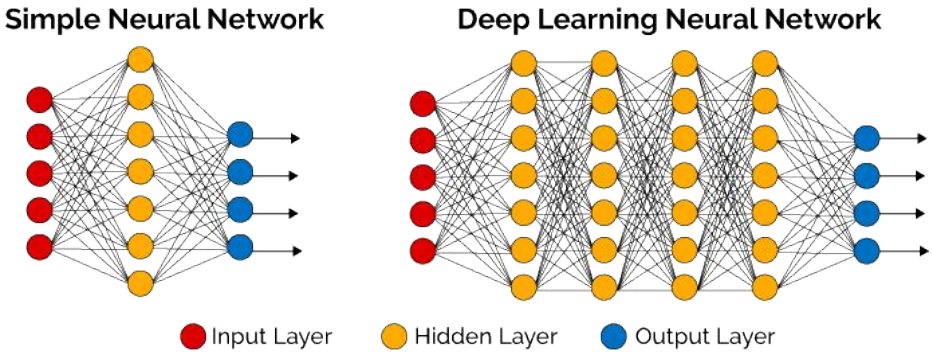
Fonte: https://www.mql5.com/pt/articles/3473

A função ReLU, comparada com as demais, permite um treinamento de uma grande base de dados de forma rápida, sendo a função mais utilizada para a maioria das aplicações.

2.1.3 Camadas

As redes neurais artificiais são compostas de uma camada de entrada, uma ou mais camadas escondidas e uma camada de saída, conforme é ilustrado na Figura 3. Uma rede neural é dita simples (*Simple Neural Network*), quando possui apenas uma camada escondida. Quando há mais de uma camada escondida, é chamada rede neural profunda (*Deep Learning Neural Network*).

Figura 3 – Rede neural simples e rede neural profunda.



Fonte: https://www.deeplearningbook.com.br/o-que-sao-redes-neurais-artificiais-profundas/

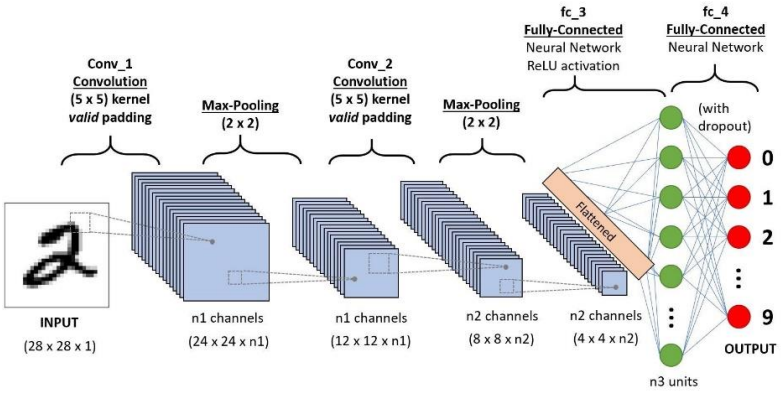
2.1.4 Processo de Aprendizagem

O algoritmo de aprendizagem é o responsável pela adaptação dos parâmetros da rede, de maneira que, em um número finito de interações do algoritmo de treinamento, haja convergência para uma solução. O treinamento de redes neurais profundas, só é possível com uma grande disponibilidade de banco de dados e uma alta capacidade de processamento de *hardware*, para lidar com uma quantidade grande de parâmetros.

2.2 REDES NEURAIS CONVOLUCIONAIS

Uma Rede Neural Convolucional ou *Convolutional Neural Networks* (*CNN*) é um algoritmo de aprendizagem profunda, capaz de captar uma imagem em sua entrada, e ao passar por várias camadas convolucionais, consegue realçar a relevância de determinadas características, como contornos, texturas entre outras características. Na Figura 4, é mostrado um exemplo de topologia de *CNN*.

Figura 4 – Exemplo de topologia de *CNN*.



Fonte: Disponível em https://www.venturus.org.br/en/machine-learning-for-laymen/

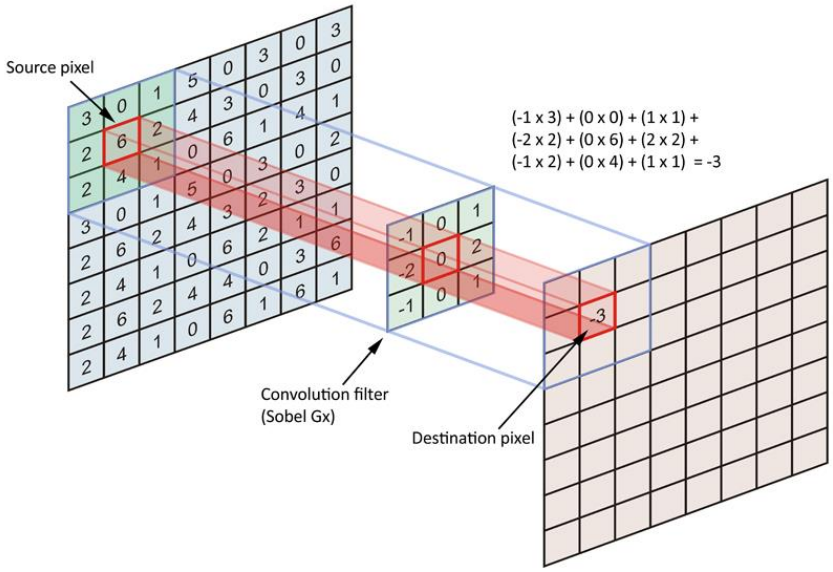
Diferentemente do processo tradicional de visão computacional, no qual o modelo inicial parte da definição de filtros a serem utilizadas, nas camadas de convolucão das *CNN's* é necessário apenas definir a quantidade, tamanhos e o passo dos filtros por camada. Não é necessário indicar qual filtro deve ser utilizado. O processo de aprendizado da rede altera os pesos ao longo do treinamento, até encontrar os melhores valores dos filtros para o conjunto de dados utilizado. (A.VARGAS; A. CARVALHO; C. VASCONCELOS, 2016).

2.2.1 Camada Convolução

A camada de convolução é a camada responsável pela extração das características da imagem. Essa extração é feita a partir da operação de convolução entre um filtro e a imagem de entrada e cada filtro aplicado extrai uma característica diferente.

Na Figura 5, é mostrado o processo de convolução em uma imagem. A imagem de entrada possui dimensões 8 x 8 e um filtro de dimensão 3 x 3.

Figura 5 – Processo de convolução



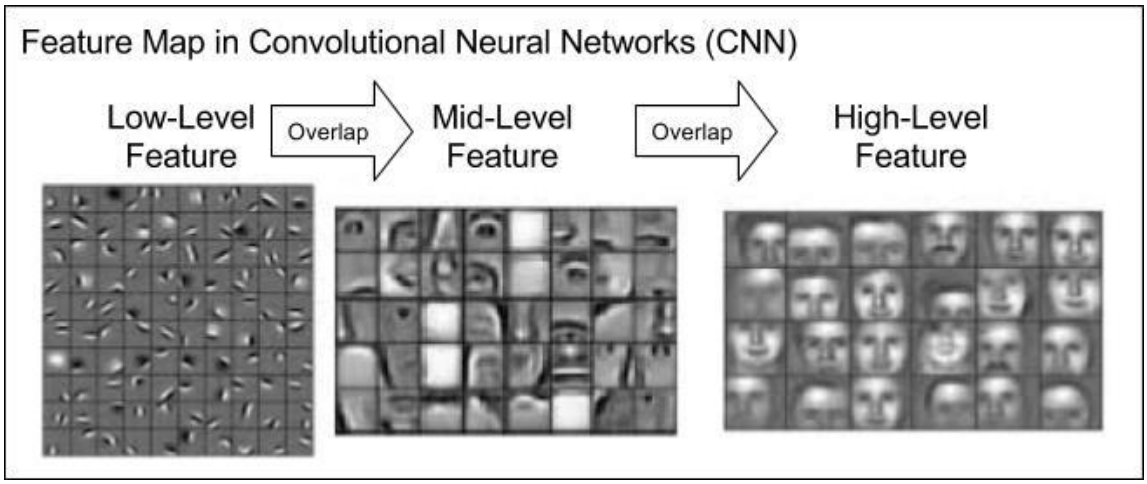
Fonte: Disponível em https://lamfo-unb.github.io/2020/12/05/Captcha-Break/

A saída da operação de convolução é obtida superpondo o filtro sobre a imagem e operando a multiplicação dos pixels em cada posição correspondentes a ambas as imagens. Esses produtos são somados e o resultado substitui o valor do pixel na imagem na mesma coordenada do pixel central do filtro. Caso se queira aplicar a operação de convolução incluindo

as bordas da imagem, é necessária a operação de *zero padding*, que é o alargamento da imagem antes do processamento.

Esse procedimento gera uma nova imagem com alguma característica destacada, seja ela uma borda, uma textura ou uma cor. Na Figura 6, pode-se observar alguns exemplos de características extraídas por diversos conjuntos de filtros.

Figura 6 – Processo de extração de características.



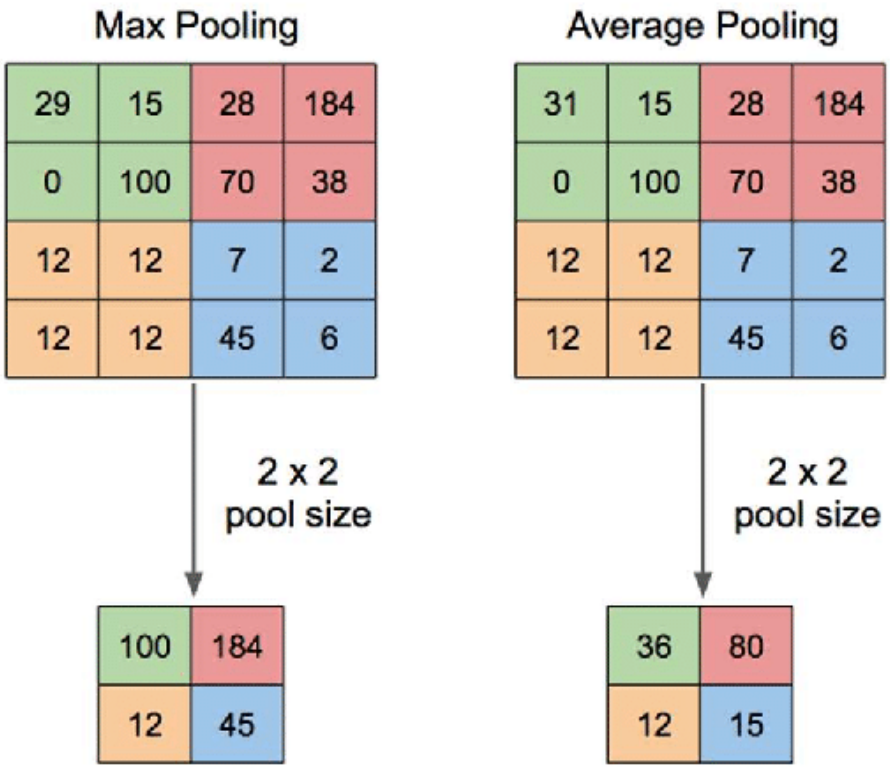
Fonte: Disponível em https://www.quora.com/How-does-a-convolutional-neural-network-recognize-an-occluded-face

2.2.2 Camada de *Pooling*

A camada de *pooling* tem a função de alterar o tamanho da imagem para fazer o processamento, comumente em *CNN's* se usa o *pooling* para reduzir o tamanho da imagem, para acelerar o processamento da rede, mas preservando as características relevantes da imagem. "O funcionamento do *pooling* é semelhante ao funcionamento da convolução: a imagem de entrada é varrida por um conjunto de neurônios. A principal diferença é que os neurônios de saída são resultado de uma função de *pooling* aplicada sobre a imagem, que pode ser a função máximo, mínimo, média, entre outras funções." (MOURA, V., 2021)

As funções de *pooling* mais utilizadas são: *MaxPooling*, substitui o valor do pixel pelo maior valor, *MinPooling*, substitui o pixel pelo menor valor e o *AverangePooling*, que substitui pelo valor médio. Na Figura 7, observa-se exemplos de *MaxPooling* e *AverangePooling.* (MOURA, V., 2021)

Figura 7 – *MaxPooling* e *AverangePooling*



Fonte: Disponível em https://www.researchgate.net/figure/Illustration-of-Max-Pooling-and-Average-Pooling-Figure-2-above-shows-an-example-of-max_fig2_333593451

2.3 CAMADA TOTALMENTE CONECTADA

A última camada da grande maioria redes neurais convolucionais é a camada responsável pela identificação dos objetos de interesse na imagem original. O vetor retornado por essa camada, em algumas arquiteturas, pode corresponder ao número de classes de um problema, sendo cada elemento do vetor a probabilidade para a imagem de entrada pertencer a uma classe. Por exemplo, se o problema é a classificação de cachorros e cabras em uma imagem, o vetor retornaria 2 valores, correspondentes a probabilidade ser um cachorro e uma cabra respectivamente.

2.4 *YOU ONLY LOOK ONCE* (*YOLO*)

A *YOLO* V4 é uma das *CNN's* de detecção, reconhecimento e classificação de objetos mais utilizada do mundo (BOCHKOVSKIY; WANG; LIAO, 2020), pois sua arquitetura é totalmente aberta e disponível ao público em geral para consumo acadêmico e comercial. Originalmente, seus modelos pré-treinados oferecem detecção para cerca de 80 classes, que

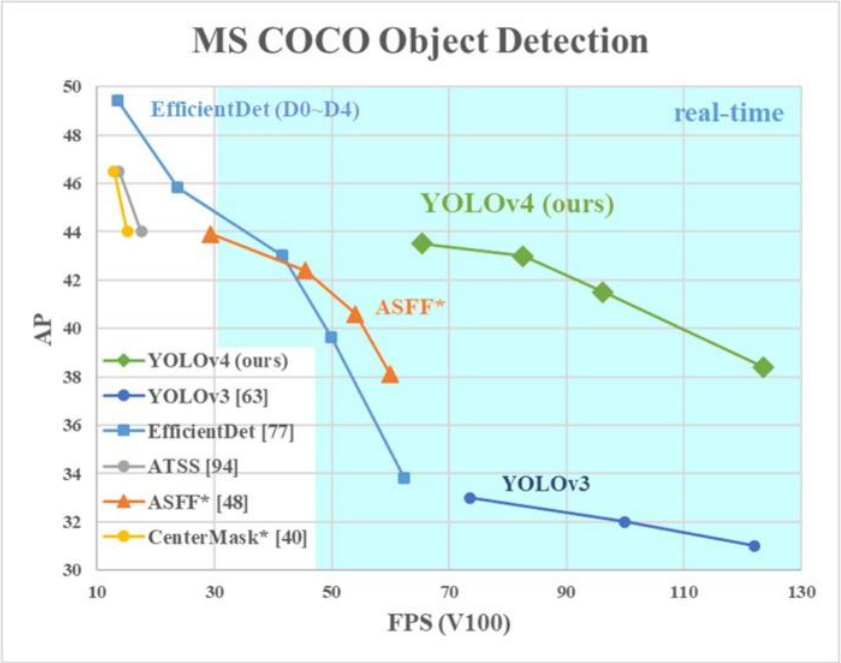
foram treinados em *datasets* com dezenas de milhões de imagens (BOCHKOVSKIY; WANG; LIAO, 2020).

Esse modelo apresenta grande acurácia para detecção de objetos dos mais diversos tipos, e oferece um grande suporte para que seja possível treinar novos modelos a partir da transferência de aprendizado dos modelos já pré-treinados (REDMON, 2016).

O grande destaque da *YOLO* em relação as redes similares é a capacidade de operar com uma quantidade muito elevada de *FPS* (*Frames Per Second*), podendo operar em tempo real a depender do hardware de vídeo disponível. Na Figura 8 é ilustrada uma comparação entre a acurácia de detecção da *YOLO* e de suas concorrentes em relação ao aumento da quantidade de *FPS* (BOCHKOVSKIY; WANG; LIAO, 2020).

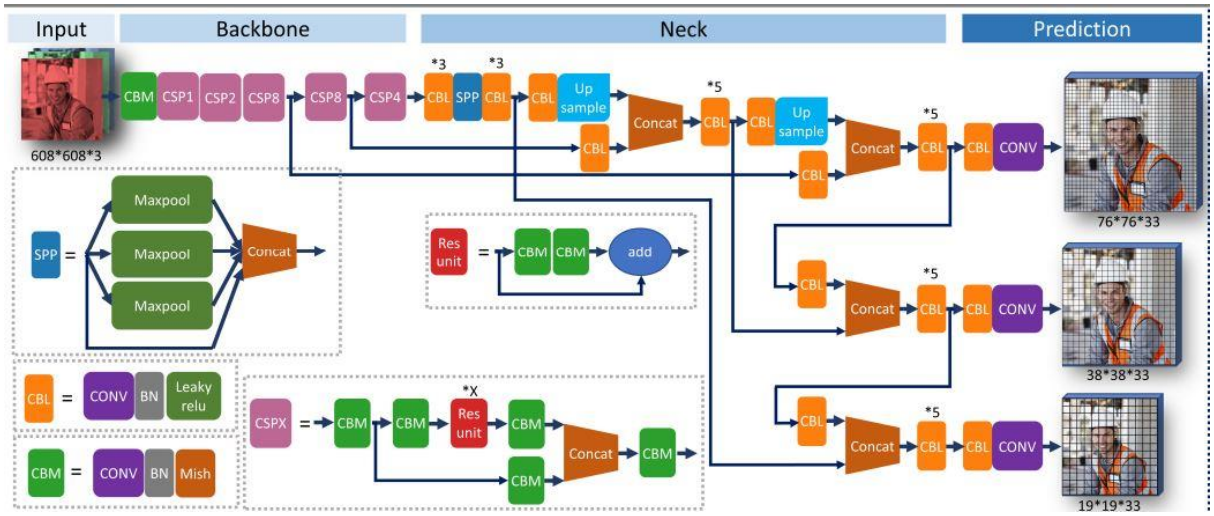
O que torna a *YOLO* tão superior as demais redes de conexão comuns é a sua arquitetura com múltiplas camadas densas, criando um *backbone* extremamente complexo, como ilustrado na Figura 10, resultando assim no contra efeito de modelos muito maiores em termos de armazenamento e arquiteturas que tendem a consumir mais memória de vídeo. (BOCHKOVSKIY;WANG; LIAO, 2020) A *YOLO* V4 é composta pela *CSPDarknet-53* (*convolutional neural network and backbone for object detection*), *SSPNet* (*Scale Selection Pyramid Network*), PANet (*Path Aggregation Network*) e 3 *YOLO* heads. (*et al*. LI, Y.; WANG H., 2020)

Figura 8 – *BenchMark YOLO* V4.



Fonte: (BOCHKOVSKIY; WANG; LIAO, 2020)

Figura 9 – Arquitetura YOLO V4



Fonte: (*et al* WANG, Z.; WU, Y., 2021)

O *backbone CSPDarknet-53* é responsável por extrair as características da imagem de entrada através de reenquadramentos. A rede contém 53 camadas convolucionais de tamanho 1x1 e 3x3. Todas as camadas convolucionais são conectadas com uma camada de normalização. Todas as funções de ativação na *YOLO* V4 são do tipo *Leaky-ReLU,* por requerer menos poder computacional. A *SSPNet* utiliza diversas camadas de *Max Pooling* de tamanhos 5, 9 e 13, e a *PANet* é utilizada para extrair características repetidamente. (*et al*. LI, Y.; WANG H.)

A função de perda da *YOLO* V4 consiste em 3 parcelas: uma relativa a existir um objeto na região, uma relativa a ser um objeto real, e outra relativa a ser um objeto de alguma classe, indicadas respectivamente nas Equações de (1) a (4). (*et al*. LI, Y.; WANG H.)

$$Loss = \lambda_1 L_{conf} + \lambda_2 L_{cla} + \lambda_3 L_{loc} \tag{1}$$

$$L_{conf} = -\sum \left(Obj_i ln(p_{ij}) + (1 - O_{ij}) ln(1 - p_{ij}) \right) \tag{2}$$

$$L_{cla} = -\sum_{i \in Box} \sum_{j \in CLass} \left(O_{ij} \ln(p_{ij}) + (1 - O_{ij}) ln(1 - p_{ij}) \right) \tag{3}$$

$$L_{loc} = 1 - IOU(A, B) + \frac{d_{AB}^2 (A_{ctr}, B_{ctr})}{l^2} + \alpha v \tag{4}$$

Em (2), Obj_i se há um objeto na caixa de predição i, e o resultado é 0 ou 1. p_i refere-se a probabilidade de haver um objeto real na caixa de predição e o seu valor é obtido calculando uma função sigmoide.

Em (3), O_{ij} e p_{ij} indicam se existe um objeto da classe *j* e a probabilidade da previsão na caixa delimitadora, respectivamente. A *YOLO* V4 adota o algoritmo *Complete Intersection Over Union* (*CIOU*) (*et al*. ZHENG. Z; WANG. P, 2020) para calcular a perda do objeto por deslocamento, mostrada em (4), levando em consideração a proporção αv e a distância euclidiana até o centro (A_{ctr}, B_{ctr}) para uma caixa delimitadora predita.

2.5 OPEN SOURCE COMPUTE VISION (OPENCV)

A *Open Source Computer Vision* ou *OpenCV* é uma biblioteca de código aberto criada para várias linguagens de programação, que permite a manipulação de imagens de forma matricial, além de disponibilizar a implementação de diversas técnicas de processamento digital de imagens e o uso de redes neurais.

2.6 LINGUAGEM *PYTHON*

Python é uma linguagem de programação interpretada e de alto nível, ou seja, roda as instruções executando uma instrução por vez (WES MCKINNEY, 2018). Para seu uso, é necessário o uso de alguma ferramenta de interpretação como o Anaconda, que foi o interpretador utilizado neste projeto. Essa é uma das linguagens de programação mais utilizadas em todo mundo e caracteriza-se por ser uma linguagem de baixa complexidade de programação (HALTERMAN, 2011).

2.7 *CUDA ENGINE* E *CUDNN*

O Compute *Unified Device Architecture* ou *CUDA*, é uma *engine* de computação desenvolvida pela *NVidia* para permitir o uso das *GPU's* de forma mais otimizada, com o auxílio dos recursos de computação paralela. Para este projeto, foi utilizado o *CUDA Toolkit* versão 11.5.

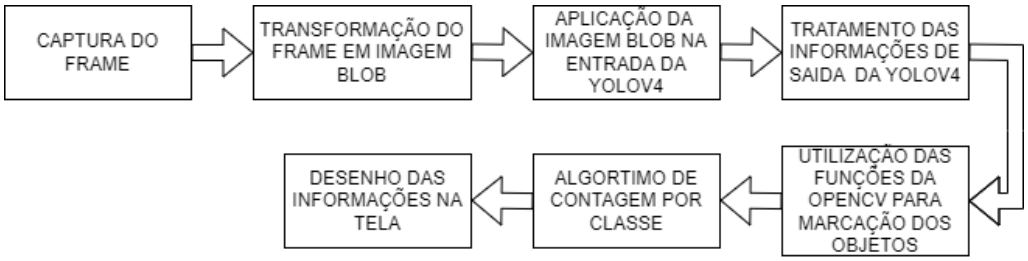
Em conjunto com o *CUDA*, é necessário instalar o *CUDNN* ou *CUDA Deep Neural Network*, que é uma biblioteca para alta performance de processamento de redes neurais profundas em *GPU's*. A versão utilizada foi a 8.5. Essa biblioteca fornece implementações altamente eficientes para convoluções, agrupamentos, normalização e camadas de ativação. Ela acelera estruturas de aprendizado amplamente utilizadas, incluindo *Caffe2*, *Chainer*, *Keras*, MATLAB, *MxNet*, *PaddlePaddle*, *PyTorch* e *TensofFlow*. Pesquisadores de aprendizado

profundo e desenvolvedores de *framewoks* em todo mundo confiam no *CUDNN* para aceleração de *GPU* de alto desempenho.

3 METODOLOGIA

Este capítulo consiste na descrição do procedimento executado para chegar ao resultado esperado. Como já descrito em capítulos anteriores, foi utilizado a biblioteca do *OpenCV* com as opções de *GPU* habilitadas, além de ser necessário instalar os drivers da *NVídia*, bem como as bibliotecas *CUDA* e *CUDNN*. A solução utilizada para analisar o fluxo de veículos nas vias é ilustrado na Figura 10.

Figura 10 – Fluxograma da solução.



Fonte: Autoria Própria.

Para simular a captura do *frame* são utilizados vídeos pré-gravados, mas pode ser feita a partir uma câmera digital de vigilância. Cada frame capturado é convertido para o formato *blob* (*binary large object*), que focam em detectar regiões na imagem que se diferem em propriedades, tais como iluminação ou cor, comparado com regiões próximas. Uma região de interesse é uma região da imagem em que algumas propriedades são constantes ou aproximadamente constantes e todos os pontos em uma região de interesse podem ser considerados, em determinados pontos de vista, similares entre si.

Em seguida, a *YOLO* irá receber as imagens *blob* em sua entrada e passar por sua estrutura convolucional, fornecendo na saída um vetor composto por uma lista das informações de cada detecção encontrada pela rede no *frame*, que são: o índice da classe pertencente, a coordenada do objeto na imagem, o comprimento e a largura da região que contém o objeto e a confiança da detecção, como demonstrado no exemplo da Tabela 1.

Tabela 1 – Exemplo do vetor de saída da *YOLO*

ClasseId	Coordenadas	Largura	Altura	Confiança
0	132, 50	100	120	0.85
2	452, 1050	80	50	0.54
3	1020, 650	20	70	0.94

Fonte: Autoria Própira.

De posse desse vetor, foi desenvolvido um algoritmo para desenhar as *bounding boxes* em cada objeto da imagem com suas respectivas informações de classe e confiança. Também foi desenhado na parte superior da imagem um painel de contagem com as classes carro, caminhão, ônibus e moto e um contador de FPS, como ilustrado na Figura 11.

Figura 11 - Detecção e representação das *bounding boxes* e dos pontos centrais

Fonte: Autoria Própria.

A próxima etapa foi calcular as coordenadas do ponto médio para cada caixa delimitadora e desenhá-lo, para servir de referência para contagem de veículos nos passos seguintes. O ponto foi calculado utilizando as coordenadas da localização, a altura e a largura da caixa fornecido pela rede, de acordo com as Equações (5) e (6):

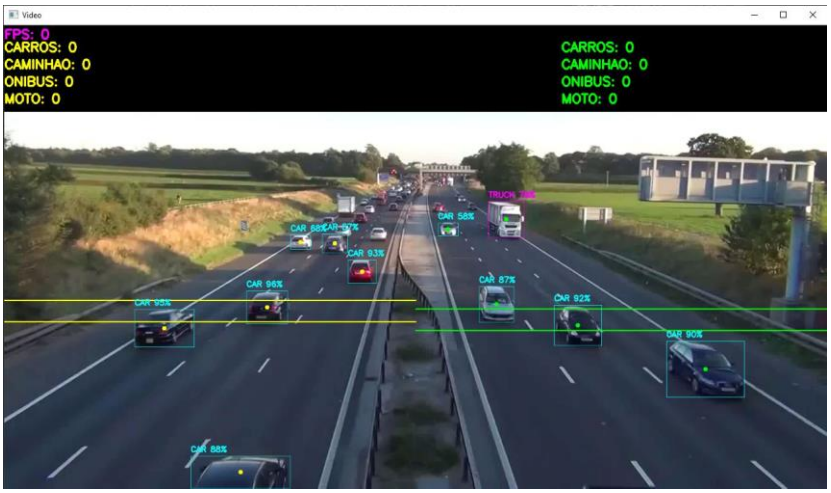
$$x_{medio} = x_{localização} + \frac{largura}{2} \tag{5}$$

$$y_{medio} = y_{localização} + \frac{altura}{2} \tag{6}$$

Uma vez calculado o ponto médio, foi marcado na imagem uma região de contagem para ambas as vias, delimitadas por retas paralelas desenhadas na imagem, como ilustrado na

Figura 12. Essas regiões foram escolhidas por serem regiões de maior estabilidade nas detecções.

Figura 12 - representação da zona de contagem



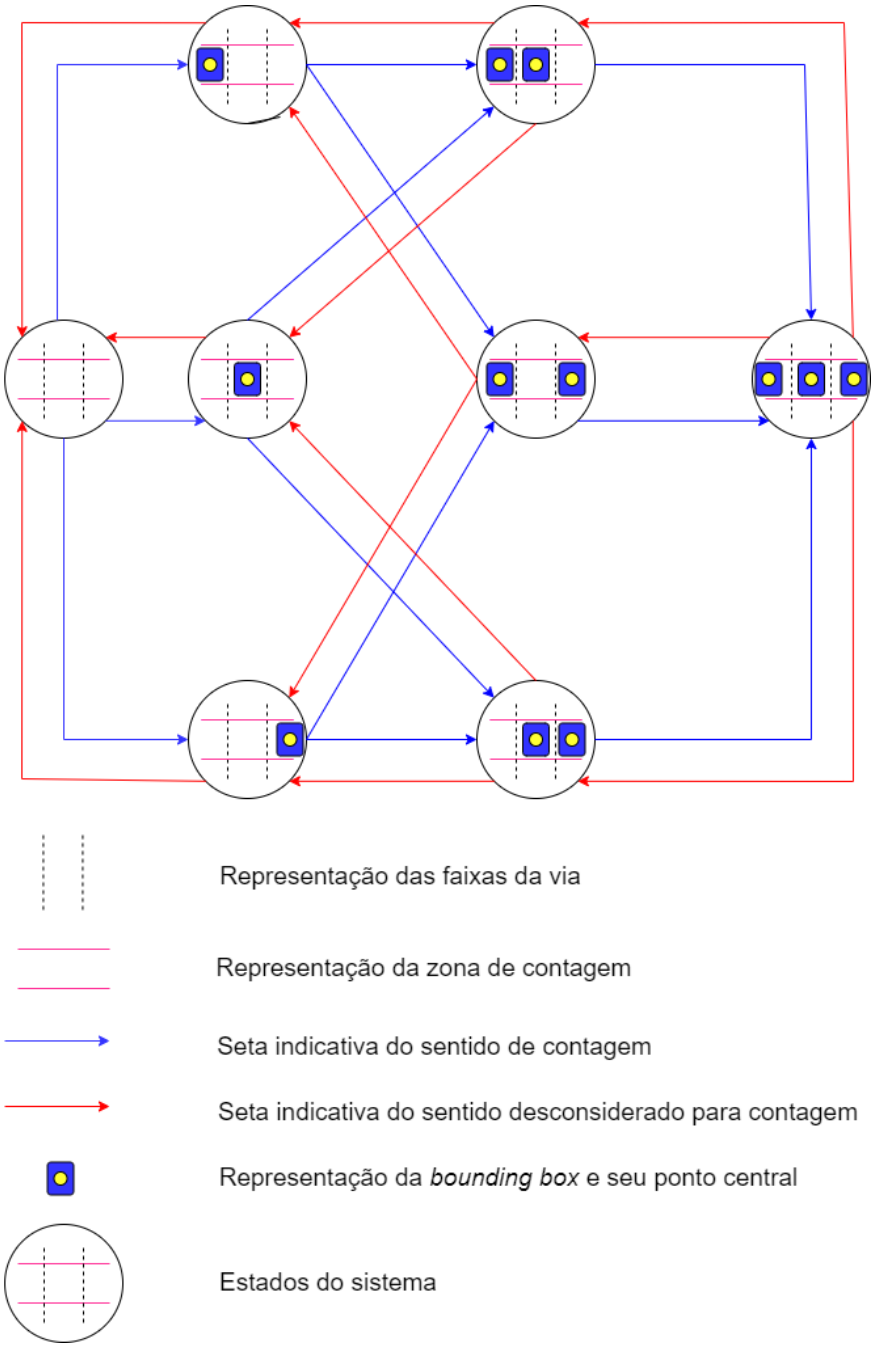
Fonte: Autoria Própria.

O algoritmo de contagem desenvolvido foi feito com uma máquina de estados finitos que considera os possíveis estados em relação a zona de contagem. A Figura 13 ilustra uma representação do algoritmo da máquina de estados implementada. Sempre que a imagem se modifica para um novo estado seguindo por uma seta azul, o algoritmo incrementa uma unidade a contagem na categoria correspondente ao veículo detectado. Caso a alteração dos estados seja feita seguindo uma seta vermelha, o algoritmo não atualiza o contador. Isso evita que a saída de um veículo da zona de contagem seja considerada uma transição válida para incremento.

O uso da máquina de estados também evita que dois veículos que entrem na zona de contagem no mesmo *frame* sejam considerados apenas um, e que um ou mais veículos parados na zona de contagem sejam contabilizadas repetidamente. Esse algoritmo foi implementado para os dois sentidos da via e funcionam de forma independente.

O algoritmo pode ser utilizado para contagens em vias em posições diferentes, fazendo-se apenas o ajuste da marcação das zonas de contagem e suas coordenadas.

Figura 13 – Algoritmo da máquina de estados.



Fonte: Autoria Própria.

4 RESULTADOS E DISCUSSÕES

Nessa seção serão apresentadas algumas métricas de desempenho para a avaliar modelos de redes neurais classificadoras. Em problemas de classificação, a avaliação das previsões realizadas pela rede, é medido pela quantidade de acertos quando a classe predita é igual ao gabarito. No caso da *YOLO*, como ela foi utilizada para classificação em vídeo, então não é possível a criação de um gabarito prévio, tornando assim o processo de avaliação empírico. Foram utilizadas duas métricas para a avaliação do desempenho da *YOLO* a acurácia e o *F1-Score*. A metodologia utilizada para a avaliação consiste nos seguintes critérios:

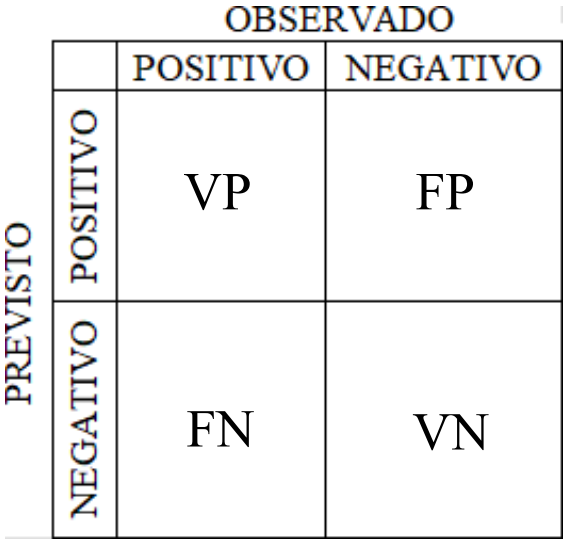
- Avaliar frames em intervalos equidistantes entre si. Neste trabalho foi utilizado um intervalo de 100 *frames*, sendo avaliados um total de 35 *frames* do vídeo.
- Foi definida uma região do *frame* onde serão consideradas detecções válidas, para padronizar a avaliação das detecções.
- Foram desconsideradas as classes ônibus e moto, devido a baixíssima frequência de aparecimento no intervalo avaliado do vídeo, impedindo assim o cálculo do *F1-Score*.
- Algumas categorias inexistentes na lista de objetos detectáveis da *YOLO* como vans e furgões foram incorporadas como corretos se detectados como a classe carro.

Também é necessário definir alguns conceitos que são utilizadas para o cálculo:

- **Verdadeiros Positivos (VP):** Número de objetos pertencentes à classe analisada que foram classificados como pertencentes a essa classe.
- **Falsos Positivos (FP):** Número de objetos não pertencentes à classe analisada que foram classificados como pertencentes a essa classe.
- **Verdadeiros Negativos (VN):** Número de objetos não pertencentes à classe analisada que foram classificados como não pertencentes a essa classe.
- **Falsos Negativos (FN):** Número de objetos pertencentes à classe analisada que foram classificados como não pertencendo a essa classe.

A partir dessas definições é possível montar uma matriz com todas as combinações denominada de matriz de confusão, que é comumente utilizada para melhor visualização nos algoritmos de classificação, como ilustrado na Figura 14.

Figura 14 – Matriz de confusão.

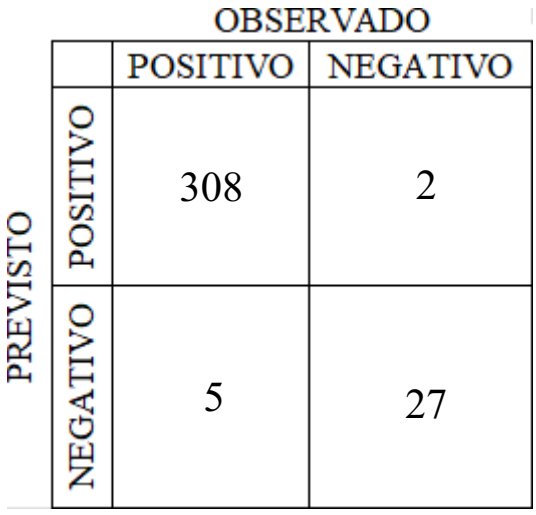


Fonte: Autoria Própria.

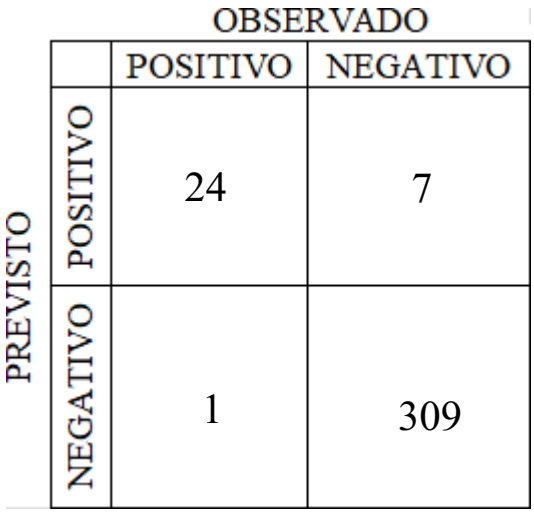
Com base nos dados obtidos com o experimento, é possível montar as matrizes de confusão, como ilustrado na Figura 15 (a) e (b), para as classes estudadas e calcular a acurácia e o *F1-Score* para cada uma delas.

Figuras 15 – Matriz de confusão para a classe carro (a) e para a classe caminhão (b)

(a)



(b)



Fonte: Autoria Própria.

Com base nos valores das matrizes, a acurácia é obtida pela Equação 7. Os dados e resultados relativos a este experimento são demonstrados na Tabela 2.

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{7}$$

Tabela 2 - Resultados de acurácia.

Classe	VP+VN	VP+VN+FP+FN	Resultado (%)
Carro	335	342	98
Caminhão	333	341	98

Fonte: Autoria Própria.

Apesar do resultado satisfatório proveniente do primeiro experimento, a fim de fazer uma análise mais apurada dos resultados obtidos, foi realizado utilizado uma segunda métrica de avaliação chamada de *F1-Score*. Essa métrica consiste no cálculo de uma média harmônica entre fatores de métricas mais precisas definidas a seguir:

- **Precisão (P):** É a proporção de verdadeiros positivos classificados como positivos. Ou seja, dentre todas classificadas como positivas, essa métrica avalia quantas são realmente positivas. Essa métrica varia entre 0 e 1 sendo 1 o valor correspondente a um desempenho ideal.

$$P = \frac{VP}{VP+FP}, 0 \leq P \leq 1 \tag{8}$$

- **Revocação (R):** É o número de de positivos classificados corretamente em relação ao total de positivos. Dentre todas as situações de classe Positivo como valor esperado, quantas estão corretas. Essa métrica varia entre 0 e 1 sendo 1 o valor correspondente a um desempenho ideal.

$$R = \frac{VP}{VP+FN}, 0 \leq R \leq 1 \tag{9}$$

O *F1-Score* é calculado combinando os resultados das equações (7) e (8) e inserindo-os na Equação (9).

$$F1\ Score = \frac{2 \cdot P \cdot R}{P + R}, 0 \leq F1\ Score \leq 1 \quad (10)$$

Com base nos valores das matrizes, os valores de *F1-Score* para cada classe e compará-los com o método de acurácia, indicados na Tabela 4.

Tabela 3 – Resultados comparativos entre Acurácia e *F1-Score*

Classe	Acurácia	F1-Score
Carro	98%	99%
Caminhão	98%	86%

Fonte: Autoria Própria.

Como é possível constatar, mesmo avaliando com um método mais apurado, o índice de acerto da *YOLO* na detecção correta de veículos é alto.

5 CONCLUSÃO

Foi apresentado neste trabalho uma solução computacional que permite a contabilidade de veículos de forma automática para análise do fluxo de trânsito. O sistema pode ser utilizado por órgãos de vigilância de trânsito. A vantagem do uso da *YOLO* se mostrou fortemente promissora com as probabilidades de detecção, classificação e contagem de veículos sem duplicidade e em tempo real, garantindo uma alta confiabilidade e uma capacidade de detecção massiva.

Em um contexto de cidades inteligentes, a grande capacidade de processamento de informações automatizada permite gerar relatórios constantes que ao longo do tempo podem indicar o comportamento da mobilidade urbana, facilitando as decisões administrativas relativas a esse setor.

REFERÊNCIAS BIBLIOGRÁFICAS

HAYKIN, S. S. Redes Neurais Princípios e Prática. 2rd ed Bookman, 2007.

VARGAS, A. C.; CARVALHO, A. M.; VASCONCELOS, C. N. Um estudo sobre Redes Neurais Convolucionais e sua aplicação em detecção de pedestres, 2016.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. YOLOv4: *Optimal Speed and Accuracy of Object Detection*. 2020.

MCKINNEY, W. *Python* para Análise de Dados tratamento de dados com pandas, *numpy* e *ipython*. 2 ed ed. Novatec Editora Limitada, 2018.

HALTERMAN, R. L. *Learning to program with python*. 2011.

GASSMANN, O.; BOHM, J.; PALMIÉ, M. *SMART CITIES Introducing Digital Innovation to Cities*. 1rd ed ed. *Emerald Publishing Limited*, 2019.

GONZALEZ, R. C.; WOODS, R. E. Digital Image Processing (3rd Edition). Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

MELO, Marcos Vinícius Lisboa. **Detecção, Contagem e Rastreio Veículos em Intersecções utilizando técnicas de Visão Computacional e Inteligência Artificial.** Alexandro José Virgínio dos Santos. 2021. TCC (Graduação) – Curso de Engenharia Elétrica, Universidade Federal da Paraíba, 2021.

REDMON, J. You only look once: Unified, real-time object detection. In: . [s.n.], 2016. Disponível em: <https://ui.adsabs.harvard.edu/abs/2015arXiv150602640R>.

FELTRIN, Fernando. Visão Computacional em Python. Ed. Uniorg.

CHOLLET, François. Deep Learning with Python. 2 ed. ed, Hanning, 2020.

Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, ``Distance-IoU loss: Faster and better learning for bounding box regression," in Proc. AAAI, 2020, pp. 12993_13000.