PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS

BRUNO ROMÃO

REDES NEURAIS CONVOLUCIONAIS PARA A DETECÇÃO DE OBJETOS

CAMPINAS

2023

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM GESTÃO DE REDES DE TELECOMUNICAÇÕES

BRUNO ROMÃO

REDES NEURAIS CONVOLUCIONAIS PARA A DETECÇÃO DE OBJETOS

Dissertação apresentada ao Programa de Pós-Graduação *Stricto Sensu* em Gestão de Redes de Telecomunicações do Centro de Ciências Exatas, Ambientais e de Tecnologias, da Pontifícia Universidade Católica de Campinas, como exigência para obtenção do título de Mestre em Gestão de Redes de Telecomunicações.

Orientador: Prof. Dr. Eric Alberto de Mello Fagotto

CAMPINAS
2023

Ficha catalográfica elaborada por Adriane Elane Borges de Carvalho CRB 8/9313
Sistema de Bibliotecas e Informação - SBI - PUC-Campinas

006.42
R761r

Romão, Bruno

Redes neurais convolucionais para a detecção de objetos / Bruno Romão. - Campinas: PUC-Campinas, 2023.

84 f.: il.

Orientador: Eric Alberto de Mello Fagotto.

Dissertação (Mestrado em Gestão de Redes de Telecomunicações) - Programa de Pós-Graduação em Gestão de Redes de Telecomunicações , Escola Politécnica, Pontifícia Universidade Católica de Campinas, Campinas, 2023.
Inclui bibliografia.

1. Processamento de imagens. 2. Redes neurais. 3. Detecção de sinais. I. Fagotto, Eric Alberto de Mello. II. Pontifícia Universidade Católica de Campinas. Escola Politécnica. Programa de Pós-Graduação em Gestão de Redes de Telecomunicações . III. Título.

23. ed. CDD 006.42

BRUNO ROMÃO

REDES NEURAIS CONVOLUCIONAIS PARA DETECÇÃO DE OBJETOS

Dissertação apresentada como exigência para obtenção do título de Mestre em Gestão de Redes de Telecomunicações ao Programa de Pós-Graduação em Gestão de Redes de Telecomunicações do Centro de Ciências Exatas, Ambientais e de Tecnologias.
Área de Concentração: Gestão de Redes e Serviços.
Orientador (a): Prof. Dr. Eric Alberto de Mello Fagotto.

Dissertação defendida e aprovada em 03 de abril de 2023 pela Comissão Examinadora constituída dos seguintes professores:

Prof. Dr. Eric Alberto de Mello Fagotto
Orientador da Dissertação e Presidente da Comissão Examinad
Pontifícia Universidade Católica de Campinas

Prof. Dr. Ademar Takeo Akabane
Pontifícia Universidade Católica de Campinas

Prof. Dr. Edson Luiz Ursini
Universidade Estadual de Campinas - UNICAMP

Dedico este trabalho primeiramente a Deus por ter sempre me ajudado
e à minha família pelo apoio em todo momento.

AGRADECIMENTOS

À Deus pela saúde, pela família e pelas oportunidades que Ele tem me proporcionado.

À minha família por oferecer todo o suporte necessário para realizar o Mestrado.

Ao Prof. Dr. Eric Alberto de Mello Fagotto pela paciência, tempo, disponibilidade, pelas valiosas orientações e aulas ministradas.

Aos Professores, Prof. Dr. Ademar Takeo Akabane, Prof. Dr. Carlos Alberto de Castro Júnior, Prof. Dra. Cecília de Freitas Morais, Prof. Dr. Frank Herman Behrens, Prof. Dra. Lia Toledo Moreira Mota, Prof. Dr. Marcius Fabius Henrique de Carvlho e Prof. Dra. Marina Lavorato de Oliveira pelas aulas ministradas.

À PUC-Campinas pela concessão da bolsa de estudos para cursar o Mestrado.

RESUMO

Carros autônomos (ACs) e sistemas avançados de assistência ao motorista (ADAS) contam com redes neurais convolucionais (CNNs) para detecção de objetos. No entanto, a degradação da imagem causada por condições climáticas adversas, como chuva, neve e neblina, pode diminuir o desempenho de uma CNN. Assim, este trabalho apresenta o desenvolvimento de uma técnica de processamento de imagem com o objetivo de mitigar tal problema. Primeiramente, após uma extensa avaliação de modelos para detecção de objetos, nossa escolha recaiu sobre a YOLOv3, devido a seu compromisso entre precisão e tempo de inferência. Posteriormente, o treinamento e teste de uma CNN YOLOv3 foi investigado para carros, semáforos, semáforos, pedestres e ciclistas/motociclistas. O desempenho foi avaliado estimando-se a precisão média e média da precisão média (mAP) para cada uma das classes de objetos mencionadas. Foi implementada uma técnica de pré-processamento baseada em OpenCV para mitigar a degradação imposta por condições climáticas adversas. Em vista disso, os filtros do OpenCV de erosão, dilatação e *joint bilateral filter* foram considerados durante o treinamento e testes dos conjuntos de dados *Berkeley DeepDrive* (BDD100K) e *Detection in Adverse Weather Nature* (DAWN). O trabalho desenvolvido apresenta os benefícios potenciais do uso de filtros OpenCV como aumento de dados durante treinamento e testes. Nossos resultados mostram uma melhora em torno de 3% no mAP durante os testes com DAWN.

Palavras-chave: Detecção de Objetos. CNN. OpenCV. Aumentação de dados. Processamento de Imagem.

ABSTRACT

Autonomous cars (ACs) and advanced driver-assistance systems (ADAS) have relied on convolutional neural networks (CNNs) for object detection. However, image degradation caused by adverse weather conditions like rain, snow, and fog can decrease the performance of a CNN. So, this paper presents the development of an image-processing technique aimed to mitigate such a problem. First, after an extensive evaluation of models for object detection, our choice fell on YOLOv3, because of its compromise between precision and inference time. Afterwards, the training and test of a YOLOv3 CNN was investigated for cars, traffic signals, traffic lights, pedestrians, and riders. Performance was evaluated by estimating the average and mean average precision (mAP) for every one of the mentioned object classes. An OpenCV based pre-processing technique to mitigate the degradation imposed by adverse weather conditions was implemented. Hence, the OpenCV filters of erosion, dilation and joint bilateral filter were considered during training and tests of the datasets Berkeley DeepDrive (BDD100K) and Detection in Adverse Weather Nature (DAWN). The developed work presents the potential benefits of OpenCV filters use as data augmentation during training and testes. Our results show an improvement around 3% in mAP during tests with DAWN.

Keywords: Object Detection. CNN. OpenCV. Data augmentation. Image Processing.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

LISTA DE ABREVIAÇÕES E SIGLAS

ADAS	*Advanced Driver-Assistance Systems*
BDD100K	*Berkeley DeepDrive*
CNN	*Convolutional Neural Networks*
DAWN	*Detection in Adverse Weather Nature*
FP	*False Positive*
FN	*False Negative*
FPN	*Feature Pyramid Networks*
GPU	*Graphics Processing Units*
JBF	*Joint Bilateral Filter*
IoU	*Intersection over Union*
IA	Inteligência Artificial
mAP	*mean Average Precision*
P	Precisão
R	Revocação
RNA	Redes Neurais Artificiais
TP	*True Positive*
UNIT	*Unsupervised image-to-image translation*
VA	Veículos Autônomos
VC	Visão Computacional
YOLO	*You Only Look Once*

1 INTRODUÇÃO

Cerca de 1,35 milhões de mortes ocorrem anualmente em acidentes automobilísticos no mundo todo (CDC, 2020). Em vista disso, diversos esforços têm sido empreendidos para preveni-las e, neste sentido, a visão computacional (VC) tem se mostrado uma das tecnologias mais promissoras.

Um processo fundamental na VC é a detecção de objetos, que é um método que envolve a localização e a classificação de objetos (p.ex., veículos, pessoas etc.). Neste método, criam-se caixas de contorno (localização) ao redor dos objetos e uma pontuação de confiança associada à classe (i.e., tipo de objeto) para cada caixa (MAHAUR et al, 2022). A VC, usualmente, envolve três etapas: a seleção informativa ou localização da região, a extração de características da seleção e, finalmente, a classificação do objeto. A seleção informativa da região é o processo de selecionar ou localizar os objetos contidos na imagem, criando uma caixa de contorno. Na extração de características, realiza-se a coleta de recursos visuais que representam a semântica do objeto. A classificação refere-se ao processo que categoriza o objeto alvo de acordo as categorias existentes (AZIZ et al, 2020).

Nas últimas duas décadas se pode identificar o progresso da detecção de objetos em dois períodos históricos: "período de detecção de objetos com métodos tradicionais – antes de 2014" e "período de detecção de objetos baseados em Redes Neurais Convolucionais (*Convolutional Neural Networks* – CNN) – após 2014". Considerando-se a detecção de objetos atual como uma revolução impulsionada pela CNN é possível observar a perspectiva de longo prazo das primeiras aplicações com a VC. A maior parte dos primeiros algoritmos para detecção de objetos foram construídos com base em definição da extração de características de forma manual, em que o desenvolvedor do algoritmo define quais características são relevantes para serem extraídas da imagem. Devido a falta de métodos de extração de características eficazes eram necessários projetos sofisticados (ZOU et al., 2023). Algoritmos pertencentes a este período são: *Viola Jones Detectors* (VIOLA; JONES, 2001), *Histograms of oriented gradientes* (*HOG) Detector* (DALAL; TRIGGS, 2005) e *Deformable Part-based Model* (DPM) (FELZENSZWALB et al, 2008).

Com a introdução das CNN e a disponibilização de vastos conjuntos de imagens para treinamento e teste a VC apresentou progresso expressivo (CYGERT; CZYZEWSKI, 2020). Sistemas de VC baseados em CNNs têm mostrado resultados promissores na

detecção de pedestres, veículos e outros objetos (HNEWA; RADHA, 2020) e atingem resultados comparáveis ou melhores que humanos (HE et al, 2015). A despeito desse cenário encorajador, Veículos Autônomos (VA) e Sistemas Avançados de Assistência ao Motorista (*Advanced Driver-Assistance Systems* – ADAS) ainda estão sujeitos a falhas importantes. Isso ocorre, principalmente, quando operam em condições climáticas adversas, em função da queda na qualidade das imagens adquiridas. A vulnerabilidade a pequenas mudanças nas imagens de entrada pode ser explicada pelo fato de que as CNNs tendem a explorar padrões de alta frequências durante o treinamento e falham quando testadas em uma mudança de distribuição dos dados. Comumente é realizada uma avaliação do modelo em imagens de diferentes distribuições ou adicionando distorções às imagens utilizadas. Deste modo, uma solução possível para mitigar os efeitos de condições adversas é a partir da utilização de técnicas de aumentação de dados (CYGERT; CZYZEWSKI, 2020; HNEWA; RADHA, 2020). A aumentação de dados consiste na prática de gerar dados sinteticamente a partir das imagens coletadas. Esses dados gerados artificialmente expandem o conjunto de imagens original usado para o treinamento. As operações nas imagens podem ser transformações como rotação ou pequenas alterações no brilho, além de outras possíveis operações que podem ser utilizadas as quais aumentam o desempenho do modelo em condições adversas (SUMMERS; DINNEEN, 2019).

Com tal cenário em vista, neste trabalho, desenvolve-se um sistema de detecção de objetos baseado na CNN YOLOv3 (REDMON; FARHADI, 2018) e com aumentação de dados a partir de recursos da biblioteca OpenCV. Para o treinamento e teste da CNN, utilizaram-se imagens do *Berkeley DeepDrive* (BDD100K) (YU et al., 2020), que são bastante diversificadas com relação às localidades e condições climáticas. Para os testes com situações climáticas adversas, considerou-se o conjunto *Detection in Adverse Weather Nature* (DAWN) (KENK; HASSABALLAH, 2020). O impacto da aumentação de dados no desempenho da CNN foi avaliado mediante a estimativa da precisão média para cada classe de objeto e a média geral das precisões médias.

1.1 OBJETIVOS

Avaliar o desempenho da CNN YOLOv3 (REDMON; FARHADI, 2018) para detecção de objetos a partir da utilização dos conjuntos de imagens BDD100K e DAWN. Adicionalmente, implementar técnicas de aumentação de dados para aumentar o desempenho na detecção de objetos em situações climáticas adversas. Desta maneira, os objetivos gerais e específicos são definidos:

1.1.1 Objetivos específicos

- Realizar o treinamento das CNNs com o conjunto de imagens BDD100K;
- Avaliar o desempenho das CNNs treinadas com imagens em climas favoráveis e em condições adversas de clima com os conjuntos de imagens BDD100K e DAWN;
- Implementar a aumentação de dados das imagens com filtros da biblioteca OpenCV e avaliar o desempenho da CNN;
- Avaliar o impacto da aumentação de dados no desempenho da CNN.

1.1.2 Objetivo geral

Contribuir para o desenvolvimento de tecnologias para visão computacional aplicada à ADAS.

1.1.3 Contribuição do trabalho

- Implementação de um sistema de VC para detecção de objetos em condições adversas com o uso da CNN YOLOv3;
- Utilização dos filtros da biblioteca OpenCV como técnica de aumentação de dados para detecção de objetos;
- Comprovação do potencial de recursos do ambiente interativa do Google Colab para fins acadêmicos na área de detecção de objetos;
- Aumento do desempenho da detecção de objetos em condições adversas a partir de técnicas de aumentação de dados com o OpenCV.

1.2 ORGANIZAÇÃO DO TRABALHO

No Capítulo 2, apresentam-se os principais conceitos relacionados à inteligência artificial e suas subáreas. No Capítulo 3, é apresentada a biblioteca OpenCV e os filtros utilizados neste trabalho. No Capítulo 4, discutem-se as métricas de avaliação de detecção de objetos consideradas. No Capítulo 5, apresenta-se a CNN Yolo-v3. No Capítulo 6, são descritos os conjuntos de imagens considerados para treinamento e testes da CNN. No Capítulo 7, é apresentada a metodologia considerada neste trabalho. No Capítulo 8,

discutem-se os resultados obtidos e, finalmente, no Capítulo 9, são apresentadas as conclusões.

1.3 REVISÃO DA LITERATURA

HNEWA e RADHA (2021) apresentam uma revisão das técnicas emergentes para mitigação da influência da chuva em detecção de objetos. Para isso, utilizaram: (i) métodos de remoção de chuva, (ii) método não supervisionado de translação de imagem para imagem (Unsupervised image-to-image translation – UNIT) (LIU; BREUEL; KAUTZ, 2017) e (iii) adaptação de domínio (CHEN et al., 2018) em conjunto com dois algoritmos de detecção de objetos, YOLOv3 (REDMON; FARHADI, 2018) e Faster R-CNN (REN et al., 2017). Os dois algoritmos de detecção de objetos foram treinados somente com imagens em condição de clima favorável. Na fase de teste, utilizaram-se, primeiramente, imagens em clima claro e, na sequência, imagens em condições de chuva. Constatou-se, no último caso, degradação no desempenho tanto da YOLOv3 como da Faster R-CNN. Entretanto, adicionando-se as técnicas de UNIT e de adaptação de domínio, mitigou-se o efeito da chuva em algumas classes, o que não aconteceu com os métodos de remoção de chuva (HNEWA; RADHA, 2021).

Imagens adquiridas em condições limitadas, como em condições adversas, podem apresentar baixa resolução e qualidade insuficiente, o que leva a generalização insuficiente do modelo e baixa robustez. Um método também utilizado para tratativa deste problema é a aumentação de dados. A essência deste método é em agregar valor aos dados existentes, sem a necessidade de coletar mais dados. Quando o conjunto de dados não é suficiente para atender um determinado objeto o método de aumentação de dados é usado para gerar novos dados similares a distribuição original. Elementos como ruídos ou imagens aleatórias são introduzidos e aumentam o desempenho do modelo. O objetivo é aumentar a capacidade de generalização sem alterar a categoria das imagens (HARIS; GLOWACZ, 2022; LIU; SU; WEI, 2022).

CYGERT e CZYŻEWSKI (2020) examinaram modelos de detecção de pedestres em condições de mundo real, com dados de testes provenientes de distribuições diferentes daquela do treinamento. Isso foi feito: avaliando-se o desempenho da CNN com imagens com condições diferentes de iluminação (p.ex., diurnas vs. noturnas) e impondo-se distorções sintéticas às imagens. Verificou-se que o desempenho do modelo base (sem

imposição de distorções sintéticas às imagens) diminuiu drasticamente nestes testes e, portanto, testar com diferentes distribuições, como no caso com imagens em diferentes condições de iluminação e com distorções sintéticas, é crucial para uma avaliação realista do modelo. Além disso, mostrou-se que a aumentação de dados melhora consideravelmente a robustez do modelo. Um novo esquema de aumentação foi proposto, que considera durante o treinamento uma imagem combinada com fragmentos estilizados da imagem original, o que melhora o desempenho do modelo na detecção de pedestres.

HARIS et al (2022) realizaram um comparativo de cinco algoritmos populares de detecção de objetos R-FCN (DAI et al., 2016), Mask R-CNN (HE et al., 2020), SSD (LIU et al., 2016), RetinaNet (LIN et al., 2020) e YOLOv4 (BOCHKOVSKIY; WANG; LIAO, 2020). Os algoritmos foram treinados e testados com o conjunto de imagens BDD100K. Foram utilizadas técnicas de aumentação de dados para o treinamento, como alteração de tamanho da imagem de forma aleatória, adição de ruído, deslocamento de imagem, desfoque, além da técnica de aumentação de dados Mosaica, que utiliza quatro imagens ao invés de apenas uma. O desempenho dos algoritmos foi avaliado a partir de métricas de precisão de precisão média (AP), média da precisão média (mAP) e tempo computacional de detecção. Mesmo com a utilização da técnica de aumentação de dados, os resultados em condição favorável de clima foram superiores àqueles obtidos em condições adversas. Os resultados experimentais apontaram que, dentre os algoritmos os que apresentaram melhor desempenho foram: YOLOv4 e Mask R-CNN. Entretanto, em termos de tempo computacional e desempenho observou-se que o algoritmo que apresentou melhor resultado foi a YOLOv4 para detecções de objetos.

LIU et al (2022) avaliaram o impacto da aumentação de dados na detecção de pedestres em ambientes noturnos. Três diferentes tipos de métodos foram utilizados: múltiplas amostras de aumentação, aumentação pelo método não-supervisionado *Generative Adversarial Network* e múltiplas amostras. Demonstrou-se que os três diferentes métodos melhoram o desempenho na detecção de pedestres em ambiente noturno.

2 INTELIGÊNCIA ARTIFICAL

A inteligência artificial (IA) é um termo que abrange uma gama de métodos e algoritmos aplicáveis à tecnologia computacional, sendo uma área de estudo no campo da ciência da computação. O foco da IA é no desenvolvimento de métodos de computação em substituições a tarefas humanas e com isso envolve atividades de aprendizado, raciocínio e autocorreção (KOK et al., 2010; VILLÁN, 2019). A utilização de recursos de IA permitem que sistemas computacionais processem informações de forma semelhantes a humanos. Dentre as subdisciplinas que estão dentro de IA estão: aprendizagem de máquina, redes neurais artificiais e redes neurais profundas (VILLÁN, 2019).

2.1 APRENDIZAGEM DE MÁQUINA

Aprendizagem de máquina pode ser definida como métodos computacionais que utilizam a experiência para realizar predições de forma mais correta e aumento de desempenho. A experiência diz respeito a dados coletados disponíveis para aprendizado. A origem dos dados pode ser de forma manual. Neste modo, uma pessoa tem a responsabilidade de coletar os dados e etiquetá-los para formar os conjuntos de dados de treinamento. O sucesso das predições depende da qualidade e quantidade dos dados coletados para o treinamento (MOHRI; ROSTAMIZADEH; TALWALKAR, 2018).

O aprendizado pode ser supervisionado, não-supervisionado e semi-supervisionado. O método supervisionado de aprendizagem relaciona os respectivos dados de saída com base em um conjunto conhecido de dados de entrada. Os dados de saída são rotulados de acordo com a classe correspondente, como uma pessoa por exemplo. Os dados de entrada e de saída são agrupados de acordo com a classe correspondente e a partir da experiência com esses dados ocorre o desenvolvimento da experiência para que as saídas sejam previstas a partir das entradas até então desconhecidas do conjunto de treinamento. O foco principal do treinamento é encontrar a relação entre os dados de entrada e de saída (BRAGA; CARVALHO; LUDERMIR, 2000; FRANCO, 2014). O aprendizado supervisionado é utilizado normalmente em tarefas de classificação e regressão (MOHRI; ROSTAMIZADEH; TALWALKAR, 2018). No caso do método não-supervisionado, apenas os dados de entrada são conhecidos (FRANCO, 2014) e os dados não são fornecidos com exemplos rotulados com a classe correspondente, ou seja, não são atribuídos rótulos para os dados de saída. A tarefa do algoritmo de aprendizado de máquina é realizar o agrupamento dos dados de entrada e descobrir padrões, sendo esses padrões denominados *clusters* (DECKER; FOCARDI, 1995; MCCALLUM; NIGAM;

UNGAR, 2000). Os métodos não supervisionados são usados em tarefas de agrupamento e redução dimensional (MOHRI; ROSTAMIZADEH; TALWALKAR, 2018). Já o aprendizado semi-supervisionado consiste em utilizar algoritmos de aprendizado que utilizam exemplos rotulados e não rotulados. A ideia principal deste aprendizado é utilizar os exemplos rotulados para guiar o processo de aprendizado com os exemplos não rotulados. Os algoritmos semi-supervisionados podem ser usados tanto em tarefas de classificação como de agrupamento (BASU, BANERJEE, MOONEY, 2022).

2.2 REDES NEURAIS ARTIFICIAIS

As Redes Neurais Artificiais (RNA) foram inicialmente propostas na década de 40 com a ideia de simular o cérebro humano para trabalhar com problemas de aprendizado (ZHAO et al., 2019). Desta maneira, as RNAs têm um papel importante para problemas de aprendizado de máquina (NILSSON, 1998). A popularidade das RNAs somente aumentou nas décadas de 80 e 90 com: (i) a introdução do algoritmo de retro-propagação (ii) a emergência de largos conjuntos de dados, como ImageNet (KRIZHEVSKY et al., 2012), (iii) o progresso computacional das unidades gráficas de processamento (Graphics Processing Units – GPU) e (iv) o desenvolvimento do design das redes (ZHAO et al., 2019).

O cérebro humano é constituído por milhões de neurônios com comunicação entre eles de forma não linear (FRANCO, 2014). As RNAs procuram modelar o cérebro humano sendo compostas por milhares de unidades de processamentos simples (neurônios). O neurônio artificial recebe várias entradas e gera uma única saída, de forma que cada neurônio é interconectado com outros neurônios em um esquema de camadas (SHARMA et al., 2012). De acordo com a entrada ou os pesos da conexão, cada neurônio produz uma ativação criando uma cadeia de ativações. Isto permite que um dado de entrada em uma camada inicial seja processado até uma camada final e uma saída desejada seja gerada (SCHMIDHUBER, 2015).

Na Figura 1, se encontra o modelo proposto por McCulloch e Pitts. O neurônio é apresentado contendo entradas (x), respetivos pesos (w), a combinação das entradas (∑) e a saída gerada. A combinação das entradas é a partir da soma do produto das entradas pelos pesos, após isto passam pela função de ativação (f(a)) que produz a saída (y).

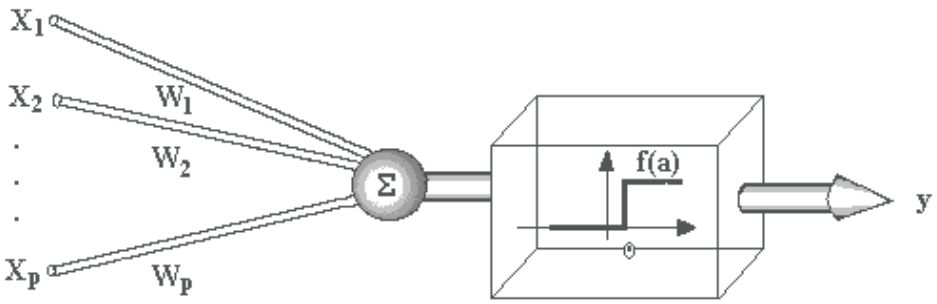

Figura 1 – Neurônio artificial de McCulloch e Pitts (CARVALHO, 2009)

O modelo de RNA Perceptron foi desenvolvido no fim da década de 1950 com intuito inicial de reconhecer padrões de fala e escrita (BRAGA; CARVALHO; LUDERMIR, 2000). Foi uma das mais básicas estruturas introduzidas nas RNAs na década de 1950, considerado como um dispositivo de tomada de decisão pela ponderação das entradas (MITCHELL et al., 1997). Para cada entrada existe um peso que considera a relevância do dado de entrada para a saída. Contudo, limitações como falta de dados de treinamento e, majoritariamente, baixa capacidade computacional resultaram em um desinteresse pela área (ZHAO et al., 2019). À medida que estas limitações foram sendo superadas, as RNAs passaram a surgir novamente.

Com o intuito de otimizar ainda mais estas redes, foram desenvolvidos algoritmos de *backpropagation*. Em um treinamento considerando o algoritmo de *backpropagation* o funcionamento da rede é a partir de duas etapas. Primeiramente, um padrão de dados é apresentado à camada de entrada. Então, estes dados são processados pela rede, camada por camada, até que a saída seja produzida pelas últimas camadas. Posteriormente a esta etapa, a saída gerada é comparada a saída desejada para os dados de entrada considerados. Caso a saída prevista não estiver correta o erro associado é calculado. Este erro é propagado a partir da camada de saída até a camada de entrada, conforme o erro é propagado os pesos das conexões das unidades das camadas internas vão sendo modificados. As redes que utilizam *backpropagation* operam com uma variação da regra delta: a regra delta generalizada. A regra delta padrão aplica um gradiente descendente no quadrado da soma do erro para funções de ativação lineares, enquanto a regra delta generalizada funciona quando são usadas na rede unidades com uma função de ativação sei-linear, que é uma função diferenciável e não decrescente (CARVALHO, 2009).

As RNAs são formadas por três camadas principais de neurônios: a camada de entrada, as camadas intermediárias e a camada de saída (Figura 2). A camada de entrada é onde os dados iniciais são inseridos. As camadas intermediárias são neurônios conectados a camada de entrada e que redefinem os valores iniciais a partir de ajustes de pesos. Na camada de saída os neurônios são finalmente ativados de acordo com o caminho usado pelas camadas intermediárias (BABA et al, 2013). A principal inovação das RNAs foi quanto ao uso das camadas intermediárias, que com o ajuste de pesos, que tornaram mais ágeis os ambientes adaptativos (FRANCO, 2014).

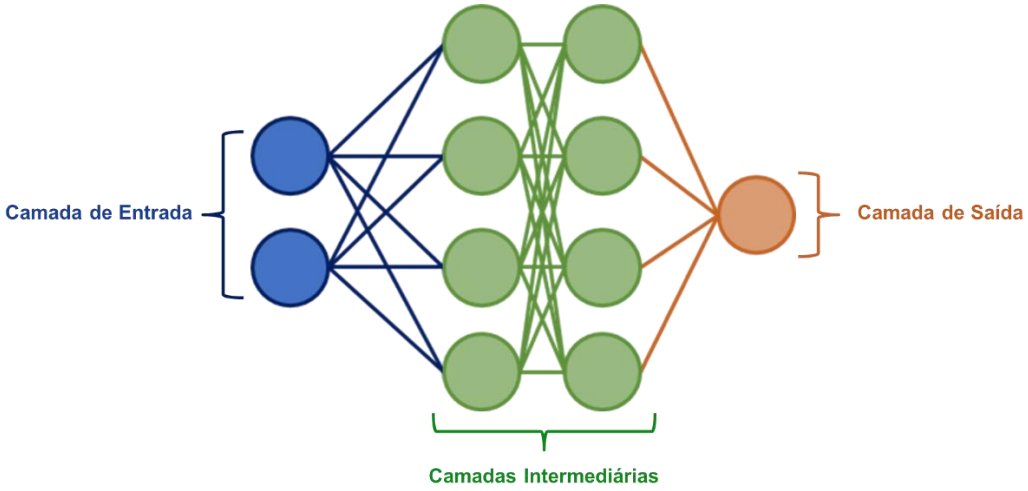

Figura 2 – Representação das camadas das RNAs (Autoria Própria)

2.3 REDES NEURAIS ARTIFICIAIS PROFUNDAS

A aprendizagem profunda e o aprendizado de máquina pertencem a área de inteligência artificial. Os algoritmos de aprendizado de máquina apresentavam limitações no processamento de dados brutos. Os dados necessitavam ser transformados em dados compatíveis com os algoritmos de forma a atender os requerimentos dos cálculos de aprendizado. O surgimento da aprendizagem profunda permitiu a inserção dos dados de forma natural, sem a necessidade de serem transformados, sendo possível a detecção automática dos dados relevantes para as tarefas desejadas (LECUN et al, 2015).

Os modelos de aprendizagem profunda, de forma geral, são redes neurais com muitas camadas, mais profundas e complexas (BENGIO, 2009). Os cálculos são acumulados em várias camadas e cada camada subsequente torna mais abstrata a representação dos dados. Isto permite que funções complexas sejam aprendidas e os

recursos menos essenciais sejam suprimidos. Adicionalmente, é possível manipular conjuntos extensos de dados, incluindo Big Data (LECUN et al, 2015).

De forma genérica, uma rede profunda do tipo *perceptron* com L camadas representa um mapeamento $f(r_o; \theta): \mathbb{R}^{N_o} \rightarrow \mathbb{R}^{N_L}$ de um vetor de entrada $r_o \in \mathbb{R}^{N_o}$ para um vetor de saída $r_L \in \mathbb{R}^{N_L}$ por meio de L etapas iterativas, de modo que (KAMASSURY, 2020):

$$r_l = f_l(r_{l-1}; \theta_l), \qquad l = 1, \dots, L \tag{1}$$

No caso de uma arquitetura com camadas totalmente conectadas, a l-ésima camada tem o mapeamento de acordo com a seguinte forma:

$$f_l(r_{l-1}; \theta_l) = g(W_l r_{l-1} + b_l) \tag{2}$$

Em que:

- $W_l \in \mathbb{R}^{N_l \times N_{l-1}}$ é a matriz de pesos referente a camada *l;*
- $b_l \in \mathbb{R}^{N_l}$ denota-se ao vetor bias;
- g (·) refere-se à função de ativação, em geral, aplicada de forma individual a cada elemento de entrada da camada, isto é $[g(z)]i = g(z)i$;
- $\theta_l = \{W_l, b_l\}$ denota o conjunto de parâmetros da camada *l.*

A Figura 3 ilustra uma arquitetura de uma rede neural profunda. Nesta arquitetura ilustrada cada neurônio recebe na entrada a contribuição de cada neurônio da camada anterior e alimenta sua saída para todos os demais neurônios da próxima camada (KAMASSURY, 2020).

A aprendizagem profunda é considerada como um aprendizado não supervisionado. Mesmo que somente os dados de entrada sejam disponibilizados é possível determinar os padrões e desenvolver habilidades de criar grupos e classes automaticamente (RUSSELL; NORVIG, 2009).

A evolução da aprendizagem profunda se dá por uma série de fatores notáveis como aprender representações de dados com vários níveis de abstração, a capacidade de aprendizado de funções complexas e de forma direta e automática lidar com dados de conhecimento mínimo do seu domínio.

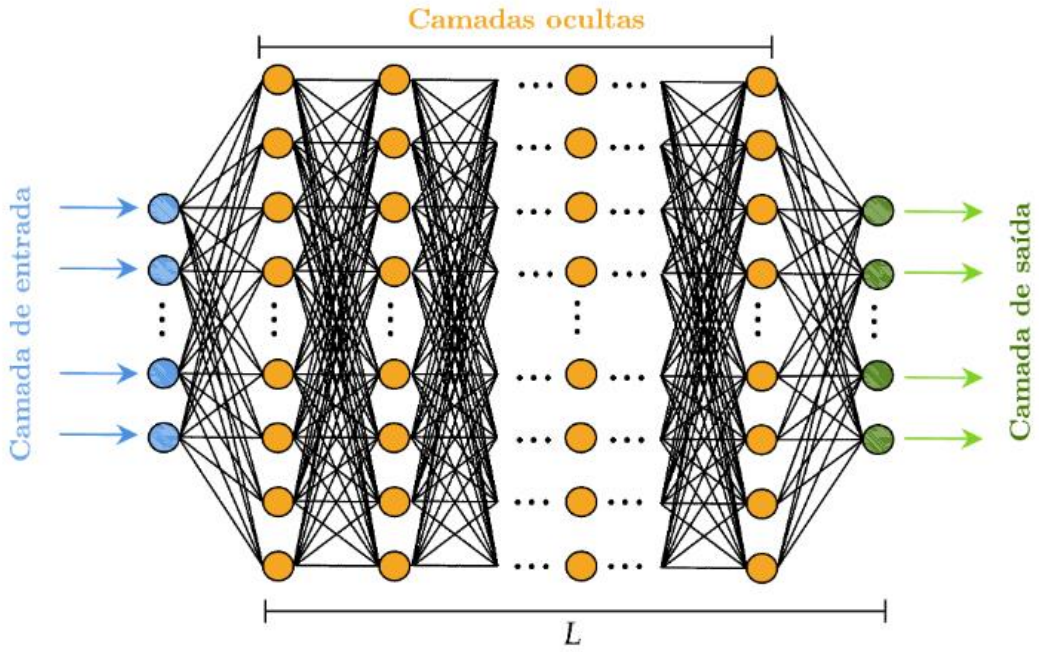

Figura 3 – Arquitetura de uma rede neural profunda do tipo *perceptron* com camadas de entrada, ocultas e de saída (KAMASSURY, 2020).

O aumento da disponibilidade de bases de dados em grande escala e o desenvolvimento computacional com a introdução de GPUs tiveram papel fundamental nos progressos da aprendizagem profunda, o que também eram fatores de impedimento da evolução das RNAs (LIU et al., 2020).

Nos últimos anos, a aprendizagem profunda promoveu diversas aplicações como reconhecimento visual, detecção de objetos, processamento de imagens, vídeos e áudios, descoberta de medicamentos e genômica (LIU et al., 2018). No que diz respeito as tarefas de entendimento da linguagem natural, se têm a expectativa da identificação de tópicos, análise de sentimentos, respostas a perguntas e a tradução de idiomas (LIU et al., 2020; LECUN; BENGIO; HINTON, 2015; BENGIO, 2009).

2.4 REDES NEURAIS CONVOLUCIONAIS

O desenvolvimento das técnicas de aprendizagem profunda, combinada à otimização das RNAs levou ao surgimento das Redes Neurais Convolucionais (*Convolutional Neural Networks* – CNN). De modo que a CNN é o modelo mais representativo da aprendizagem profunda (ZHAO et al., 2019). As CNNs são compostas por camadas qualificadas para identificar padrões mais relevantes para a rede. As camadas

aplicam a operação de convolução ao invés da matriz de multiplicação (GOODFELLOW et al, 2016; DATA SCIENCE ACADEMY, 2019). A aplicação das CNNs aparece em diversas áreas como detecção de objetos, classificação de vídeo, rastreamento de objetos, segmentação e super-resolução (SZEGEDY et al., 2015).

As CNNs são compatíveis com o processamento de matrizes múltiplas. Dentre as principais categorias de dados, estão as matrizes unidimensionais de sinais e sequências, como séries de dados temporais e de linguagem, matrizes bidimensionais de áudios e imagens e matrizes tridimensionais como imagens volumétricas e vídeos (LECUN; BENGIO; HINTON, 2015).

A CNN consiste na combinação de camadas de redes neurais, tipicamente formada por três estágios: convolução, função de ativação não-linear e camadas de subamostragem (*pooling*).

Como exemplo, na Figura 4, a partir de uma imagem como dado de entrada, a camada convolucional aplica filtros convolucionais, denonimados *kernels*, à imagem e o resultado é uma nova imagem, porém com suas dimensões relativamente reduzidas. Esta nova imagem, representada na forma de um tensor, é chamada de mapa de características (GOODFELLOW et al, 2016).

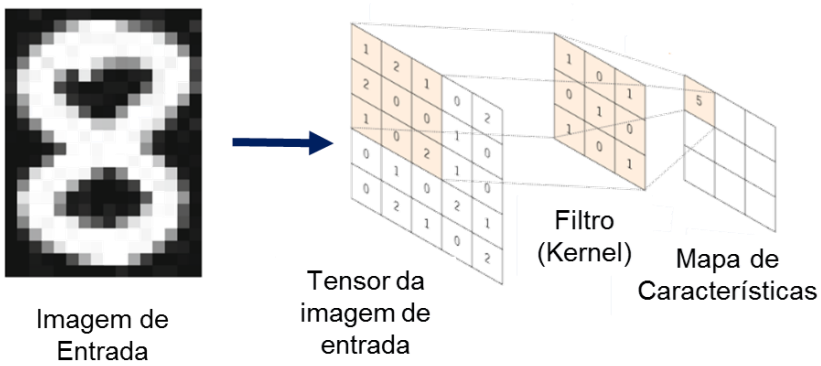

Figura 4 – Camada de convolução da CNN – Adaptado de (PATIL; RANE, 2021)

No segundo estágio, a ativação linear resultante da etapa de convolução passa por funções não lineares. A introdução da não-linearidade no modelo permite um melhor gradiente de propagação. Dentre as funções não-lineares mais utilizadas está a ReLU (Rectified Linear Unit – ReLU) (BROWNLEE, 2019a). A função ReLU retorna 0 se recebe um valor negativo e assume o valor de entrada caso seja positivo, de acordo com a Equação (3) (GOODFELLOW et al, 2016):

$$f(x) = \max(0, x) \quad (3)$$

O último estágio tem a função de resumir estatisticamente as características mais relevantes de acordo com as saídas vizinhas da camada anterior. Como demonstrado na Figura 5, considerando uma camada de "*max pooling*", somente os valores máximos do Tensor de Entrada são considerados mais relevantes e conservados para a próxima camada no Tensor de Saída (GOODFELLOW et al, 2016).

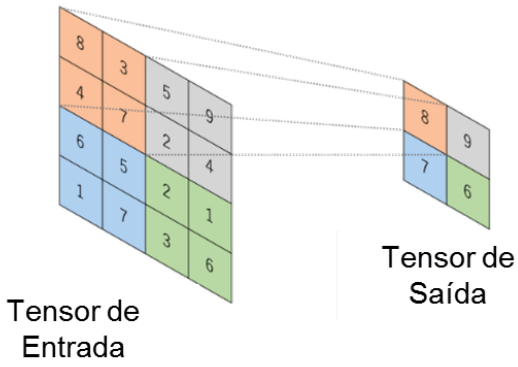

Figura 5 – Subamostragem (pooling) com valor máximo – Adaptado de (PATIL; RANE, 2021)

Na Figura 6, se encontra a topologia de uma CNN com as camadas convolucionais e as camadas de subsmostragem. As últimas camadas são totalmente conectadas com os neurônios provenientes da camada de subamostragem. Esta é apenas uma representação da topologia da CNN, visto que, em sua implementação, ela é constituída por várias camadas alternadas.

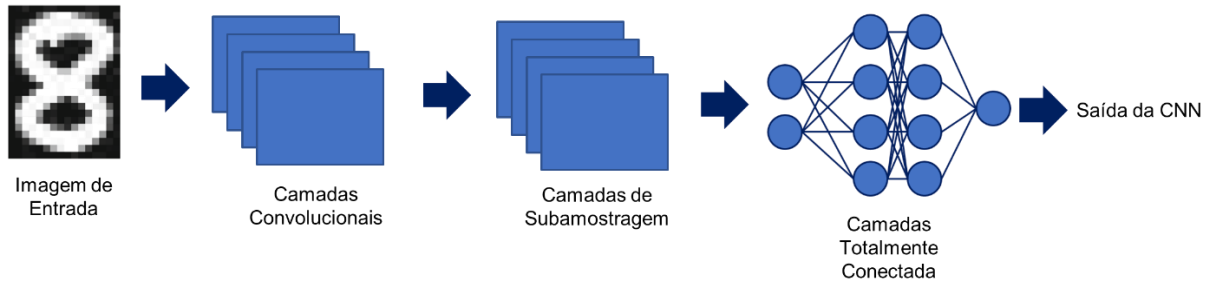

Figura 6 – Topologia de uma CNN – Autoria Própria

2.5 DETECÇÃO DE OBJETOS

Uma das aplicações mais estudadas na área de aprendizagem profunda é a de visão computacional (VC) (BALLARD; HINTON; SEJNOWSKI, 1983). A tarefa da VC é mimicar o olho humano e os componentes do cérebro que têm a função do senso de visão. A VC é muito ampla, englobando várias maneiras de processamento de imagens e vídeos, com o intuito de identificar feições, alocar classes e até o desenvolvimento de novas habilidades visuais (GOODFELLOW et al, 2016). Dentre as tarefas regulares da VC estão: classificação de imagens, classificação e localização e a detecção de objetos.

A tarefa de classificação tem o papel de predizer o que a imagem significa como um todo a partir de um conjunto de categorias pré-existentes. A imagem pode ter vários elementos, entretanto o resultado da classificação é predizer se contém o elemento esperado. A Figura 7 apresenta a classificação da categoria ‘esquilo’ a partir da imagem de entrada, apesar de conter outros elementos na imagem (SILVA, 2021).

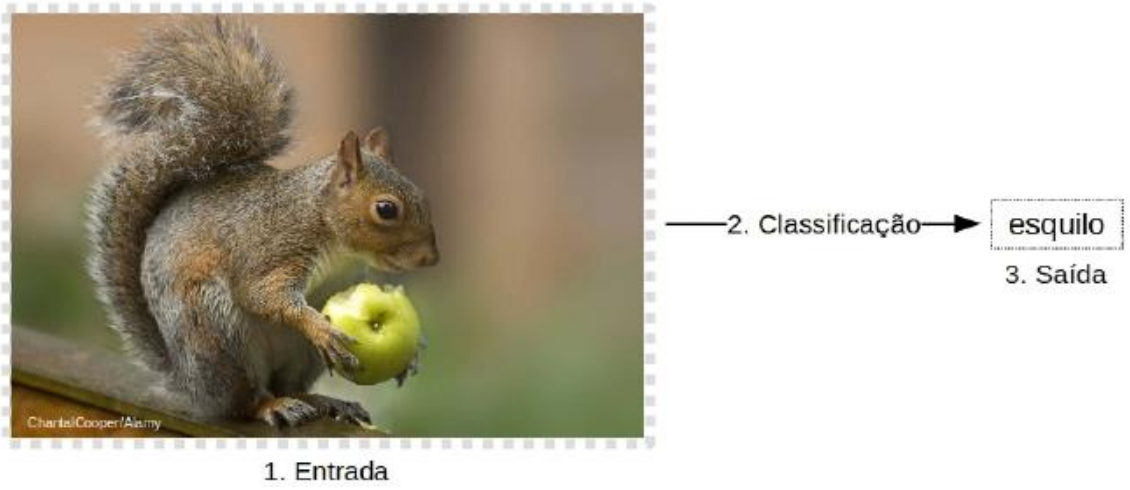

Figura 7 – Classificação de uma imagem (SILVA, 2021)

Na localização de objetos, um objeto por imagem é identificado em uma categoria e a localização deste objeto é informada pelo algoritmo. Como identificado na Figura 8, a imagem de entrada é processada pelo algoritmo e a saída é a classe do objeto ‘esquilo’ e a sua localização determinada a partir do centro da caixa de contorno vermelha prevista, que é definido por ‘x’ e ‘y’ e a largura e a altura definidas por w e h, respectivamente (SILVA, 2021).

Figura 8 – Classificação e localização de uma imagem (SILVA, 2021)

A detecção de objetos possibilita a classificação e a localização de mais de um objeto por imagem. Na Figura 9, encontra-se a detecção dos objetos das classes ‘esquilo’ e ‘maçã’ e as suas respectivas localizações a partir das predições do centro e largura a altura das caixas de contorno (SILVA, 2021).

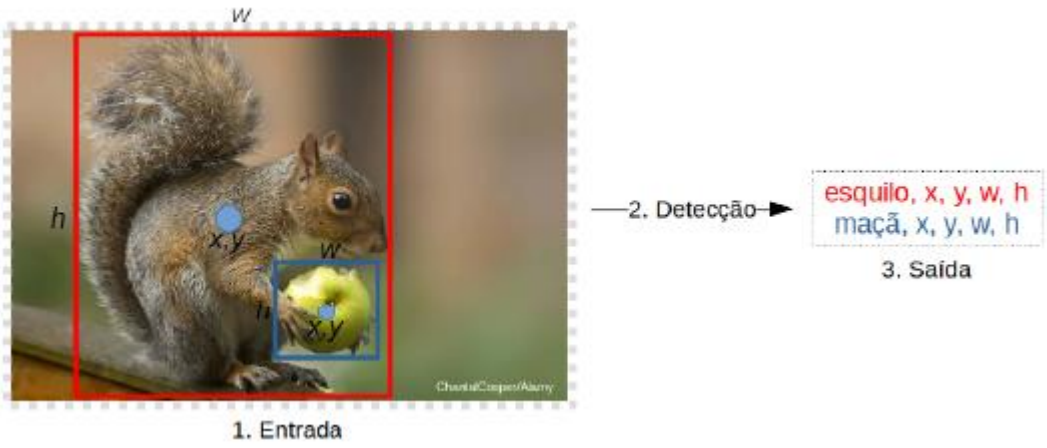

Figura 9 – Detecção de objetos (SILVA, 2021)

3 OPENCV

OpenCV é uma biblioteca que foi criada para o processamento de imagens em tempo real. Pelo fato de ter sido escrita de forma otimizada em C/C++, permite o processamento paralelo em vários núcleos, por exemplo, como em uma GPU. A sua utilização é gratuita tanto para fins acadêmicos e comerciais a partir de uma licença BSD. Este tipo de licença se aproxima do domínio público, pois é uma licença de código aberto que possui baixas restrições quanto ao seu uso (MARENGONI; STRINGHINI, 2010; VILLÁN, 2019; OPEN SOURCE, 2023)

Desenvolvida pela Intel, a biblioteca OpenCV possui mais de 500 funções, sendo dividida em cinco grupos: processamento de imagens, análise estrutural, análise de movimento e rastreamento de objetos, reconhecimento de padrões e calibração de câmera e reconstrução 3D. O objetivo da sua idealização foi de tornar a VC acessível a usuários e programadores, estando disponível com o código fonte e os executáveis otimizados para os processadores Intel, fornecendo VC em tempo real. As aplicações são diversas como: sistema de reconhecimento facial, reconhecimento de gestos, interface entre homem e máquina, realidade aumentada, detecção de distração ou fatiga, entre outras aplicações (MARENGONI; STRINGHINI, 2010; VILLÁN, 2019).

Uma das formas de processamento de imagem com a OpenCV é a partir da utilização de filtros. Os filtros podem ser aplicados no domínio do espaço ou então no domínio da frequência, quando. A imagem é filtrada no domínio da frequência e transformada novamente para o domínio do espaço (MARENGONI; STRINGHINI, 2010).

Os processos no domínio espacial podem ser formalmente descritos como (MARENGONI; STRINGHINI, 2010):

$$g(x,y) = T(f(x,y)) \tag{4}$$

Em Equação (2), f(x,y) é a imagem original, T(.) se refere ao processo de transformação da imagem e g(x,y) a imagem depois do processo de transformação. A operação T é definida sobre uma vizinhança de influência do pixel localizado na posição específica (x, y). A vizinhança pode ser compreendida como composta pelos *pixels* ao redor da posição (x, y). Estas regiões podem ser quadradas ou retangulares e também ser chamadas de máscaras (MARENGONI; STRINGHINI, 2010).

Uma aplicação bastante útil é a transformação de uma imagem em

monocromática, utilizando um valor de corte (k). Esta transformação é definida em Equação (5) (MARENGONI; STRINGHINI, 2010):

$$g(x,y) = \begin{cases} 1, se\ f(x,y) \geq k \\ 0, caso\ contrário \end{cases} \quad (5)$$

A Figura 10 ilustra a aplicação desta função para um valor de k=84. Esta técnica também é utilizada para isolar objetos de interesse na imagem. Ao lado esquerdo a imagem de um urso preto, ao lado direito uma imagem monocromática, destacando o urso. Utilizando k=84 (MARENGONI; STRINGHINI, 2010).

Figura 10 – Aplicação da transformação monocromática (MARENGONI; STRINGHINI, 2010)

3.1 FILTRO DE DILATAÇÃO

Com a imagem A e o denominado elemento refletido de estrutura $\hat{B}$, a dilatação de A por B, denotada por $A \oplus B$ é definida pela Equação (6):

$$A \oplus B = \{z \mid [(\hat{B})_z \cap A] \subseteq A\} \quad (6)$$

O elemento refletido de estrutura $\hat{B}$ é a reflexão das coordenadas do elemento de estrutura B. Se B é definido em um objeto de imagem com coordenadas (x,y), o elemento $\hat{B}$ tem as coordenadas (-x,-y). A dilatação de A por B é o conjunto de todos os deslocamentos, z, tais que B e A se sobrepõem em pelo menos um elemento (GONZALEZ, WOODS, 2002). De forma simplificada, o resultado da operação de dilatação é o valor máximo de todos os *pixels* da vizinhança para o pixel de saída. Esta operação faz os objetos mais visíveis e as linhas ficarem mais espessas (OPENCV, 2022; MATHWORKS, 2022).

Na Figura 11 ilustra-se o processo de dilatação em uma imagem de entrada (A) em escala de cinza. O elemento de estrutura (B) neste caso é definido em um formato

retangular com três colunas e seu centro é na segunda coluna. A dilatação então preserva o maior valor do pixel da sobreposição de B em A e o substitui na imagem de saída (OPENCV, 2022).

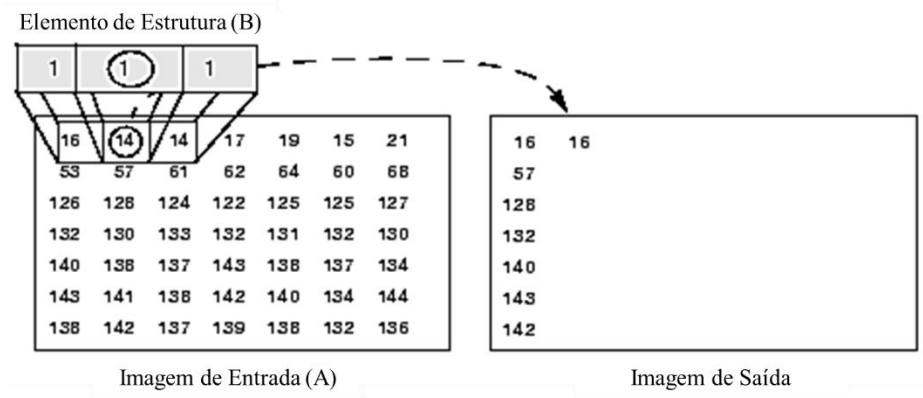

Figura 11 – Aplicação do filtro de Dilatação – Adaptado de (MATHWORKS, 2022)

3.2 FILTRO DE EROSÃO

Com a imagem A e o denominado elemento de estrutura B, a erosão de A por B, denotada por $A \ominus B$ é definida pela Equação (7):

$$A \ominus B = \{z \mid (B)_z \subseteq A\} \tag{7}$$

Esta equação indica que a erosão de A por B é o conjunto de todos os pontos z tais que B, trasladado por z, é contido em A. O elemento B é escaneado sobre a imagem A, e é computado o menor valor do pixel sobreposto por B e substitui-se esse valor no pixel da imagem em que se encontra o centro de B. O resultado desta operação é a remoção de *pixels* flutuantes e linhas mais finas para que apenas objetos relevantes permaneçam na imagem (GONZALEZ, WOODS, 2002; OPENCV, 2022).

Na Figura 12 ilustra-se o processo de erosão em uma imagem de entrada (A) em escala de cinza. O elemento de estrutura (B) neste caso é definido em um formato retangular com três colunas e seu centro é na segunda coluna. A dilatação então preserva o menor valor do pixel da sobreposição de B em A e o substitui na imagem de saída (OPENCV, 2022).

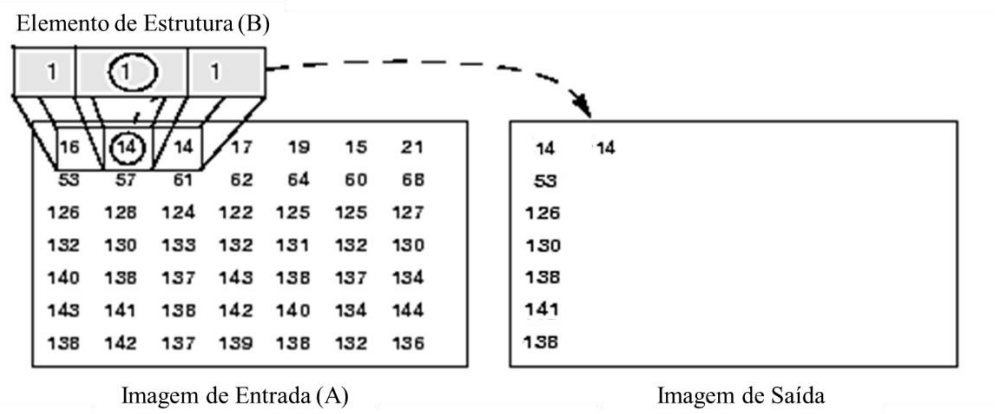

Figura 12 – Aplicação do filtro de Erosão – Adaptado de (MATHWORKS, 2022)

3.3 JOINT BILATERAL FILTER (JBF)

O filtro bilateral tem a função de realizar a média de *pixels* que estão espacialmente próximos e possuem intensidades similares. O filtro bilateral usa um kernel no domínio espacial e um kernel de intervalo avaliado a partir dos valores dos *pixels*. Para uma posição *p*, tem-se o resultado do filtro conforme Equação (8) (TOMASI; MANDUCHI, 1998):

$$J_p = \frac{1}{k_p} \sum_{q \in \Omega} I_q f\big(||p - q||\big) g\big(||Ip - Iq||\big) \tag{8}$$

Em que f é o kernel do filtro espacial, como um Gaussiano centrado sobre *p*, g é o kernel do filtro de intervalo, centralizado no valor da imagem em *p*, Ω é o suporte espacial do kernel *f* e k_p é um fator de normalização fator a partir da soma dos pesos do filtro f · g. As bordas são preservadas desde que o filtro bilateral f · g assuma valores menores conforme maior diferença de intensidade e diferença espacial entre os *pixels* vizinhos.

A partir do filtro bilateral se teve a introdução do JBF em que a aplicação do filtro de intervalo é realizada em uma segunda imagem $\tilde{I}$. Esta aplicação é utilizada, por exemplo, para combinar altas frequências de uma imagem a baixas frequências de outra (KOPF et al., 2007; PETSCHNIGG et al., 2004), de acordo com a Equação (9):

$$J_p = \frac{1}{k_p} \sum_{q \in \Omega} I_q f\big(||p - q||\big) g\big(||\tilde{I}p - \tilde{I}q||\big) \tag{9}$$

4 MÉTRICAS DE AVALIAÇÃO

Em algoritmos de detecção de objetos a localização pode se dar a partir da previsão de uma caixa de contorno. Para avaliar quão similar a caixa prevista está com relação à caixa de referência, é utilizada a métrica de avaliação da Intersecção sobre União (*Intersection over Union* – IoU). A IoU descreve a sobreposição da caixa prevista com relação à caixa de referência, onde está exatamente localizado o objeto, de acordo com a Figura 13.

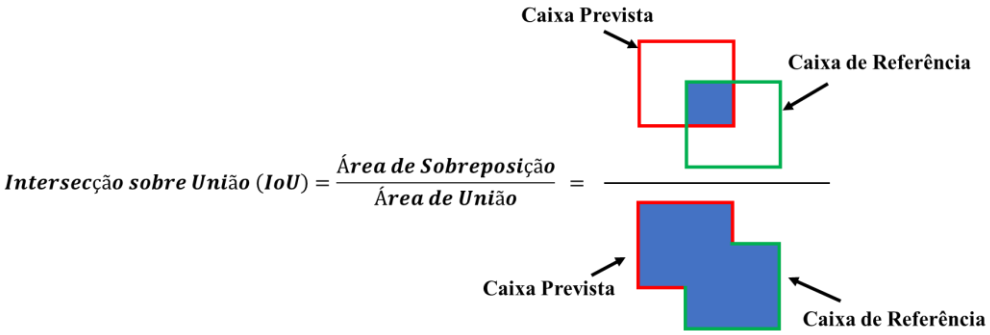

Figura 13 – Intersecção sobre União

Logo, a IoU é o resultado da divisão da área de intersecção entre a caixa prevista e a de referência pela área de união destas mesmas caixas.

São consideradas detecções verdadeiras positivas (*True Positive* – TP) se o valor de IoU > 0,5 e, caso contrário, falsas positivas (*False Positive* – FP). Se o objeto não for detectado sobre a caixa de contorno de referência, ou seja, IoU = 0, então é uma detecção falsa negativa (*False Negative* – FN) (YU; JI, 2022).

A partir disto, é possível avaliar de forma mais consistente o desempenho do algoritmo. A métrica de precisão (P), de acordo com Equação (10), avalia quão preciso é o algoritmo com relação às detecções positivas, a partir da porcentagem de detecções TP em relação ao total de detecções positivas, TP e FP. Ou seja,

$$P = \frac{TP}{TP + FP} \tag{10}$$

A métrica revocação (R) é definida como

$$R = \frac{TP}{TP + FN} \tag{11}$$

e quantifica a razão entre as detecções positivas encontradas e o total real de eventos positivos (i.e., TP+FN).

Para uma avaliação conjunta das métricas P e R, calcula-se precisão média (*average precision* – AP)

$$AP = \int_0^1 P(R)dR \tag{12}$$

A métrica de AP é calculada por integração numérica a partir da área da curva P por R, considerando IoU de 0,5 de acordo com (12). A AP foi definida originalmente no desafio de Classes Visuais de Objetos de PASCAL em 2012 para avaliar os métodos de detecção de objetos (M. EVERINGHAM, L. V. GOOL, C. WILLIAMS, J. WINN, 2012). Estes valores de AP são calculados para cada classe de objeto. Uma métrica também utilizada para avaliação do desempenho dos algoritmos é a média da precisão média (mean average precision – mAP), de acordo com (13).

$$mAP = \frac{1}{n}\sum_{i}^{n} AP_i \tag{13}$$

A mAP será calculada a partir da AP, para as n classes de objetos do algoritmo de detecção.

A YOLO conta com a função *loss* que permite avaliar os erros relacionados à localização, classificação e probabilidade condicional de cada classe ou da não existência de objeto. Esta função foi utilizada para a avaliação dos resultados de treinamento. Decidiu-se adotar uma convenção mais usual do que concorrentemente encontrada na YOLO e a função se encontra em (14):

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B} a_{ij}^{obj}[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \tag{14}$$

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} a_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] +$$

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} a_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} a_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2 +$$

$$\sum_{j=0}^{S^2} a_i^{obj} \sum_{c \, \epsilon \, classes} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

- a_i^{obj} se há objeto na célula *i* então é 1, caso contrário é 0 e não é considerada;
- a_{ij}^{obj} se a caixa de contorno *j* é responsável por predizer o objeto da célula *i* então é 1, caso contrário é 0 e a parte dependente não é considerada;
- a_{ij}^{noobj} indica um não-objeto na célula *ij*;
- B é o número de caixas de contorno previsto para a célula da grade;
- S^2 dado que a imagem de entrada é dividida em grade S x S, então para cobrir todas as células da grade tem-se S².

A primeira somatória se refere à média dos erros quadrados do centroide previsto $(\hat{x}_i, \hat{y}_i)$ em relação ao centroide espero (x_i, y_i) das caixas de contorno de cada célula.

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} a_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \tag{15}$$

A segunda somatória se refere à média dos erros quadráticos da altura e largura previstas $(\hat{w}_i, \hat{h}_i)$ com relação às esperadas (w_i, h_i) das caixas de contorno de cada célula.

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} a_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \tag{16}$$

A terceira se refere à classificação dos objetos, resultando na somatória dos erros quadráticos das probabilidades de existência de um objeto em cada célula. A

probabilidade de existência de um objeto é calculada de acordo com (18) a partir da multiplicação da probabilidade de existência de um objeto pela IoU.

$$\sum_{i=0}^{S^2}\sum_{j=0}^{B} a_{ij}^{obj}\left(C_i - \hat{C}_i\right)^2 \tag{17}$$

$$C = \Pr(Objeto)\,.\,IoU \tag{18}$$

A quarta se refere à classificação dos não-objetos, resultando na somatória dos erros quadráticos das probabilidades de não existência de um objeto em cada célula.

$$\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} a_{ij}^{noobj}\left(C_i - \hat{C}_i\right)^2 \tag{19}$$

A quinta e última se refere à somatória dos erros quadráticos das probabilidades condicionais de existir um objeto de uma determinada classe *c*.

$$\sum_{j=0}^{S^2} a_i^{obj} \sum_{c\ \epsilon\ classes} \left(p_i(c) - \hat{p}_i(c)\right)^2 \tag{20}$$

Também foi utilizada como métrica de avaliação a medida-F. De acordo com SASAKI (2007) esta medida é definida como a média harmônica da precisão (P) e revocação (R) conforme (21):

$$F = \frac{2.P.R}{P+R} \tag{21}$$

Quando se considera métricas de proporções, no caso precisão e revocação, a média harmônica é mais adequada do que a média aritmética. Em um sistema de reconhecimento de impressão digital, por exemplo, em com precisão de 1,0 e revocação de 0,2 somente 20% das impressões digitais registradas são cobertas, ou seja, o desempenho é baixo e não eficiente. A média aritmética de 1,0 e 0,2 é 0,6 enquanto a média harmônica é 0,33, o que é mais adequado para interpretar o desempenho do sistema (SASAKI, 2007).

5 YOU LOOK ONCE (YOLO)

Seres humanos possuem a capacidade de olhar um objeto e instantaneamente reconhecer qual é o objeto e onde ele está. O sistema de visão humano é rápido e preciso, o que permite a realização de tarefas complexas como a direção de veículos. Métodos de detecção de objetos que sejam rápidos e precisos permitem a direção autônoma sem sensores especializados assim como sistemas robóticos responsivos (REDMON et al., 2016).

Métodos de detecção de objetos baseados em proposta de região primeiro realizam a predição de potenciais caixas de contorno e então é realizada a classificação destas caixas propostas. Depois da classificação, um processamento adicional é necessário para refinar as caixas de contorno prevista e eliminar duplicações. Este processamento complexo é devagar e difícil de ser otimizado, pois cada componente é treinado separadamente (REDMON et al., 2016).

Observando esta lacuna nos métodos de detecção de objetos REDMON et al (2016) reestruturam a detecção de objetos como um problema de regressão e criaram a arquitetura You Look Only Once (YOLO), baseada em uma CNN. O processamento é realizado diretamente dos *pixels* da imagem, são previstas as caixas de contorno e as probabilidades das classes criando um sistema que "vê apenas uma vez" uma imagem e identifica quais objetos estão presentes e onde estão. Contribuindo desta maneira para aplicações que necessitam de rapidez e precisão. Na Figura 14, se encontra um esquemático do modelo de detecção de objetos implementado na arquitetura YOLO.

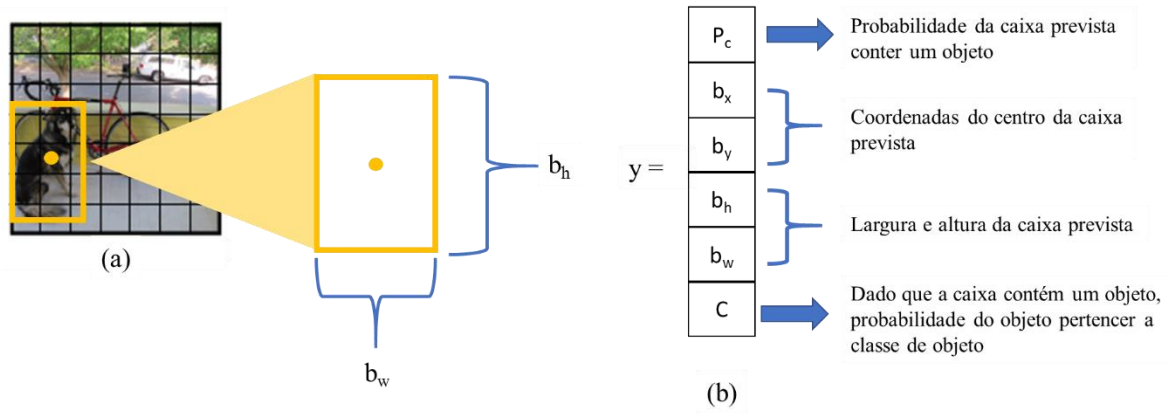

Figura 14 – (a) grade com S x S células sobre a imagem de entrada. (b) tensor de saída gerado a partir das células de grade –YOLO (REDMON et al., 2016)

Primeiramente, a imagem de entrada é dividida em uma grade de tamanho S x S, de acordo com a Figura 14 (a). A CNN predirá B caixas de contorno que, eventualmente, poderão cobrir mais do que uma célula da grade. Identificado o centro de um objeto, em dada célula, esta será responsável por sua detecção (Figura 14 (b)). Para a caixa de contorno são previstas as coordenadas de centro (b_x, b_y) e largura (b_w) e altura (b_h). Serão estimadas a probabilidade Pc da caixa conter o objeto e probabilidade C de uma célula conter o objeto da classe dado que a caixa de contorno contém um objeto. Estas informações todas estarão contidas em um tensor y, que será utilizado para a identificação do objeto. Este tensor é de dimensões $S \times S \times (5B + C)$. A constante 5 indica a quantidade de variáveis (P_c, b_x, b_y, b_w, b_h) para cada caixa de contorno B. Soma-se C pelo fato da quantidade de caixas de contornos ser independente do número de classes (SILVA, 2021).

Entretanto, a arquitetura inicial da YOLO apresentava algumas limitações, especialmente, na detecção de objetos pequenos, com relação ao tamanho das caixas, e de diferentes classes. Isso acontecia pelo fato de cada célula de grade podia conter somente duas caixas de contorno e apenas uma classe de objeto (REDMON et al., 2016). Com o objetivo de flexibilizar e aumentar a capacidade da CNN, na YOLOv2 (REDMON; FARHADI, 2017) implementaram-se as caixas de ancoragem (REN et al., 2017). Estas são caixas de altura e largura pré-definidas para capturar objetos de cada classe. Na YOLO, apenas 98 caixas de contorno eram permitidas por imagem e, com a introdução das caixas de ancoragem, tornou-se possível haver mais de mil caixas (REDMON; FARHADI, 2017). Na YOLOv3 (REDMON; FARHADI, 2018), foram combinadas camadas de redes neurais residuais (HE et al., 2016) e a predição de caixas de contorno em três níveis de escala (REDMON; FARHADI, 2018) a partir da extração de características por redes neurais piramidais (*Feature Pyramid Networks* – FPN) (LIN et al., 2017). Com as atualizações realizadas na arquitetura da YOLOv3, tornou-se possível detectar objetos complexos de diferentes tamanhos e categorias, herdando as características superiores das arquiteturas antecessoras, mantendo-se o compromisso entre tempo de processamento e precisão (TURAY; VLADIMIROVA, 2022).

5.1 YOLOV2

O foco principal desta versão foi em melhorar a revocação e erros de localização enquanto mantinha acurácia na classificação (REDMON; FARHADI, 2017). A versão da YOLOv2 chegou a apresentar resultados competitivos com rapidez quando comparada a

outros métodos robustos como Faster R-CNN (REN et al., 2017) com ResNet (HE et al, 2016) e SSD (LIU et al, 2016). Atingiu estado-da-arte tanto no desafio PASCAL VOC (EVERINGHAM et al., 2010) quanto COCO (LIN et al., 2014).

Algumas alterações foram realizadas na YOLO (REDMON et al., 2016) visando um método de detecção com melhor desempenho o que levou a YOLOv2. Foram feitas as seguintes alterações:

- A normalização de *batch* na YOLOv2 descartou a necessidade de método de regularização (IOFFE; SZEGEDY, 2015) e melhorou a rede em convergência trazendo mais estabilidade durante o treinamento. Esta implementação aumentou em mais de 2% a métrica de mAP (REDMON; FARHADI, 2017; VALIATI, 2019).
- Na primeira versão da YOLO a resolução inicial da rede era 224 x 224 *pixels* para treinamento da classificação e para treinamento da detecção a resolução se alterava para 448 x 448 *pixels*. Enquanto na YOLOv2 a resolução inicial da rede já se inicia com 448 x 448 *pixels* por 10 épocas para treinamento da classificação, isto dá tempo a rede de ajustar os filtros para trabalhar com uma entrada com resolução maior. Com esta alteração a métrica de mAP aumentou cerca de 4% (REDMON; FARHADI, 2017; VALIATI, 2019).
- As previsões das caixas de contorno eram realizadas diretamente pelas CNNs na versão inicial da YOLO, mas (REN et al 2017) demonstraram que era uma maneira ineficiente. Na YOLOv2, utilizaram-se as técnicas de caixas de ancoragem da Faster R-CNN (REN et al, 2017). O uso das caixas de ancoragem para cada localização na imagem permite que a rede apenas faça a predição de deslocamentos das caixas de ancoragem ao invés das coordenadas, o que é uma tarefa mais fácil para a predição. A métrica de revocação aumentou, enquanto a acurácia sofreu uma diminuição negligenciável (REDMON; FARHADI, 2017; VALIATI, 2019).
- Ao invés de utilizar o mesmo método de caixas de ancoragem da Faster R-CNN na YOLOv2, as dimensões das caixas de ancoragem são calculadas a partir do agrupamento por *k-means* (MACQUEEN, 1967) sobre a anotação de referência dos conjuntos de imagens de treinamento. Isto permitiu definir a priori quais dimensões das caixas de ancoragem melhor se adequariam ao conjunto de

imagens, ao invés de defini-las manualmente (REDMON; FARHADI, 2017; VALIATI, 2019).

- Restringir a previsão da localização na YOLOv2 com o uso das caixas de ancoragem às células da grade tornou o aprendizado mais estável, melhorando em 5% de acurácia em relação ao método sem a melhoria implementada (REDMON; FARHADI, 2017; VALIATI, 2019).
- A YOLOv2 realiza a detecção de objetos a partir de um mapa de características de 13 x 13 *pixels*, o que é suficiente para objetos grandes. Para melhorar a detecção de objetos pequenos foi proposta a adição de uma "camada de passagem" que recebe características de camadas anteriores da rede com mapas de características em uma resolução de 26 x 26 *pixels*. Esta camada permitiu informações mais robustas de objetos pequenos concatenando as características de baixa resolução com alta resolução. Com essa expansão de mapas de características de forma mais refinada trouxe uma melhoria de 1% em desempenho (REDMON; FARHADI, 2017; VALIATI, 2019).
- Para aumentar o desempenho da YOLOv2 em imagens de diferentes tamanhos, o tamanho da imagem de entrada é alterado de forma aleatória a cada 10 *batches* durante o treinamento. As imagens são redimensionadas em múltiplos de 32, com no mínimo 320 x 320 *pixels* e no máximo 608 x 608 *pixels*. Esta estratégia faz com que o aprendizado da rede utilize diferentes dimensões de tamanho de imagem de entrada. Isso leva a melhores detecções em diferentes resoluções (REDMON; FARHADI, 2017; VALIATI, 2019).

5.2 YOLOV3

Desenvolvida em 2018, a YOLOv3, apresentou uma série de alterações visando melhorar a precisão na detecção (REDMON; FARHADI, 2018). Nesta versão, três escalas diferentes são previstas para objetos pequenos, médios e grandes. A detecção é realizada a partir de núcleos de detecção com mapas de características de três tamanhos diferentes, em três locais diferentes da rede com *strides* de 32, 16 e 18. Para uma imagem de 416 x 416 *pixels*, os mapas de características gerados são 13 x 13, 26 x 26, 52 x 52 *pixels* para objetos grandes, médios e pequenos, respectivamente, de acordo com a Figura 15 (REDMON; FARHADI, 2018; ERAZO, 2021).

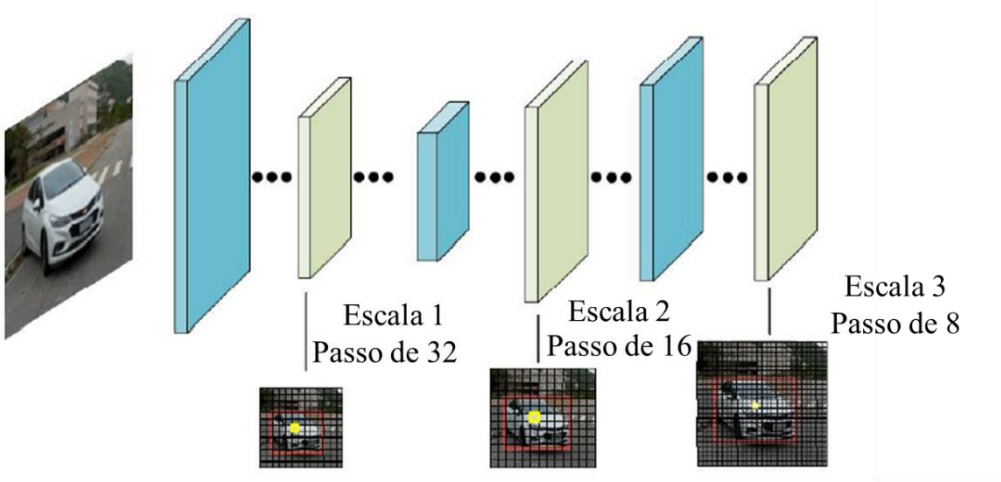

Figura 15 – Diagrama simplificado da extração de características da YOLOv3 – adaptado de (ERAZO, 2021).

Na YOLOv3 a predição da caixa de contorno, bem como na YOLOv2, é realizada a partir das caixas de ancoragem que têm tamanho similar ao dos objetos. Entretanto, na YOLOv3, ao invés de apenas 5 caixas de ancoragem por célula de grade, são usadas 9 caixas, 3 para cada escala. Com esta nova configuração na YOLOv3, podem ser previstas 10.647 caixas de contorno, enquanto na YOLOv2 somente podem ser previstas 845 caixas (REDMON; FARHADI, 2017; VALIATI, 2019).

Na YOLOv2, a arquitetura da rede era composta por 30 camadas, enquanto na YOLOv3 a rede é composta por 106 camadas. Devido ao aumento das camadas na rede a YOLOv3 tornou-se mais lenta por consequência, contudo ainda rápida o suficiente para ser comparada aos demais métodos de detecção concorrentes (REDMON; FARHADI, 2017; VALIATI, 2019).

Outra modificação, foi quanto ao uso da classificação de múltiplas de classes. A função *softmax* foi substituída por classificadores logísticos independentes, visto que a camada *softmax* impõe que cada célula da grade possua apenas uma classe, o que não ocorre em uma aplicação real. A utilização de classificadores independentes atribui um nível de probabilidade para cada classe de objeto dentro da imagem. Como exemplo, na Figura 16, com o uso dos classificadores independentes, uma rede treinada para detectar carros, vans e caminhões pode realizar a predição das probabilidades do objeto pertencer a classe 1 – carros, classe 2 – vans e classe 3 – caminhões ao mesmo tempo (REDMON; FARHADI, 2018; ERAZO, 2021).

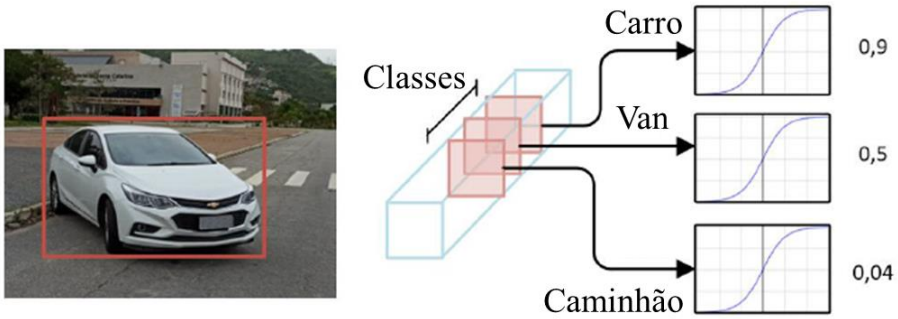

Figura 16 – Representação do classificador logístico utilizado na YOLOv3 (ERAZO, 2021)

A YOLOv3 foi avaliada no desafio de detecção COCO (LIN et al., 2014) e atingiu resultados comparáveis a métodos de detecção como RetinaNet e SSD (Figura 17). Os resultados da YOLOv3 quando comparados à SSD possui mAP semelhante, porém é cerca de 2,5 vezes mais rápida (REDMON; FARHADI, 2018; ERAZO, 2021).

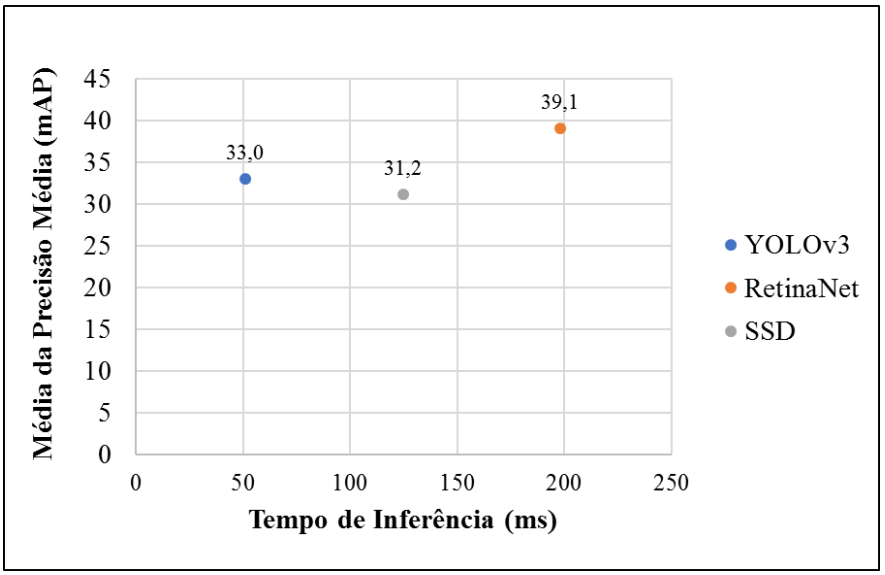

Figura 17 – YOLOv3 em relação a RetinaNet e SSD no desafio COCO (REDMON; FARHADI, 2018)

6 CONJUNTOS DE IMAGENS

Para treinamento da CNN YOLOv3 fez-se necessário o uso de conjuntos de imagens para treinamento e testes para verificação do desempenho.

Os conjuntos precisam além de possuir imagens contendo os objetos em questão para identificação (carros, sinais de trânsito, semáforos, pessoas e ciclistas/motociclistas) necessitam de anotações devidamente realizadas. O depositório utilizado para o treinamento e testes da YOLOv3 foi o *Berkeley DeepDrive* (BDD100K) (YU et al., 2020), descrito na seção 6.1.

Para avaliação do desempenho da YOLOv3 com imagens de qualidade diferente a utilizada durante o treinamento utilizou-se outro depositório de imagens, o *Detection in Adverse Weather Nature* (DAWN) (KENK; HASSABALLAH, 2020), descrito na seção 6.2.

6.1 BERKELEY DEEPDRIVE (BDD100K)

Dentre as várias bases de dados (GEIGER et al., 2013; NEUHOLD et al., 2017; ZHANG; BENENSON; SCHIELE, 2017) utilizadas para o treinamento de CNNs, apenas uma delas, a Berkeley DeepDrive (BDD100K) (YU et al., 2020) contém imagens em condições climáticas diversas devidamente anotadas.

A anotação das imagens é um processo manual, portanto, longo e custoso se se considera que, normalmente, são necessárias pelo menos dezenas de milhares de imagens para o treinamento e o teste de uma CNN.

O BDD100K é um conjunto de imagens de grande diversidade. Ele foi construído a partir de vídeos de milhares de motoristas patrocinados pela Nexar (NEXAR, 2022). No total, são 100.000 vídeos de direção coletados em mais de 50.000 trajetos, cobrindo Nova York, São Francisco e outras regiões dos Estados Unidos. Contém diversas cenas com áreas residenciais, rodovias e, além disso, foram gravadas em diferentes condições de clima (tempo claro, neve, chuva etc.). No décimo quadro do vídeo, é anotada a descrição da imagem para a detecção (YU et al., 2020). São disponibilizadas 70.000 imagens para treinamento, 10.000 imagens para validação e 20.000 imagens para testes. Pelo fato de as imagens para testes não possuírem anotação, consideraram-se as imagens de validação para a fase de testes. Dentre as classes disponíveis de objetos no BDD100K, quatro delas contém os objetos mais críticos para veículos autônomos (HNEWA; RADHA, 2021): carros, pessoas, semáforos e sinais de trânsito. Adicionalmente, usou-se a classe de

ciclistas/motociclistas para treinamento e testes, a despeito de o número delas ser comparativamente menor com o de outras classes mencionadas. Na Figura 18 se encontram algumas imagens típicas presentes no depositório BDD100K.

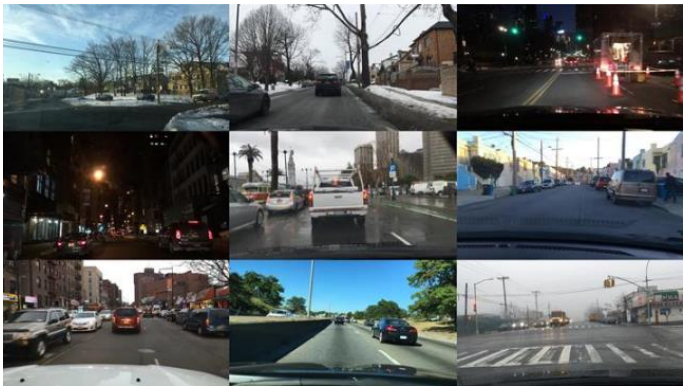

Figura 18 – Imagens típicas do depositório BDD100K

6.2 DETECTION IN ADVERSE WEATHER NATURE (DAWN)

O DAWN é um conjunto de imagens em diversos ambientes de tráfego, contendo imagens urbanas e rodoviárias com variedade de fluxos de tráfego (KENK; HASSABALLAH, 2020).

O objetivo deste conjunto é, principalmente, em fornecer dados para a investigação do desempenho dos sistemas de detecção de veículos em diferentes condições climáticas. Contém variações significativas relacionadas a veículos como: categoria, orientação, iluminação, posição e oclusão. As cenas de trânsito contêm imagens em tráfego normal e congestionado, além de rodovias e estradas urbanas.

As imagens são originadas de pesquisas no Google e Bing com pesquisas de imagens com as palavras-chave: nevoeiro, neblina, névoa, clima de inverno, tempo tempestuoso, batidas de neve pesada, chuva de granizo, tempestade de areia, tempestade de poeira, clima perigoso, clima adverso, tráfego, estradas, veículo). Então, são selecionadas e filtradas por humanos em ciclos. Todas devem respeitar os termos de uso do Google, Bing e Ficklr com a licença: “Livre para compartilhar e usar”.

São 1000 imagens em condições reais de tráfego divididas em: névoa, neve, chuva e tempestade de areia. As anotações foram realizadas a partir do LabelMe (RUSSEL et al, 2008) para cinco tipos de veículos (carros – 82,21%, ônibus – 2,05%, caminhões – 8,22%, motocicletas e bicicletas – 1,36%) e com a presença de humanos (6,07%) a partir de

anotações de ciclistas e pedestres. Na Figura 19 se encontram algumas imagens em condições de névoa, chuva, neve e tempestade de areia do conjunto de imagens.

Figura 19 – Imagens do depositório DAWN (KENK; HASSABALLAH, 2020).

7 METODOLOGIA

Durante o treinamento são ajustados os pesos da CNN YOLOv3 e essencialmente é necessária a configuração dos hiperparâmetros de treinamento, o conjunto de imagens e o ambiente utilizado. Tais configurações possuem influência no tempo de treinamento e desempenho da CNN. Na seção 7.1, se encontram as configurações utilizadas neste trabalho para o treinamento.

Após realizar-se o treinamento é imprescindível a execução de testes de verificação de desempenho da CNN. Para isso são considerados conjuntos de imagens não utilizados durante o treinamento e a definição de hiperparâmetros da CNN, os quais também influenciam no desempenho da CNN. Além disso, são consideradas métricas de avaliação usualmente consideradas em desafios de detecção de objetos. Na seção 7.2, discutem-se os detalhes considerados para os testes da CNN.

7.1 TREINAMENTO DO MODELO

Durante o treinamento da CNN YOLOv3, utilizou-se a plataforma do Google Colab. O Google Colab pertence à área de pesquisas científicas da Google e é um produto da Google Research. O Google Colab possibilita a execução de códigos arbitrários em *python* pelo navegador. Seu uso é adequado para fins educativos, análise de dados e aprendizado de máquina (GOOGLE, 2023).

Para o treinamento da CNN é necessária a configuração de um arquivo .cfg em que se definem os hiperparâmetros. Pelo fato de o objetivo do trabalho ser em avaliar o desempenho da CNN YOLOv3 e implementar técnicas de aumentação de dados para aumentar o desempenho na detecção de objetos em situações climáticas adversas não se realizou o ajuste fino dos hiperparâmetros durante o treinamento. VALIATI (2029) realizou o esforço de atualizar os hiperparâmetros da CNN YOLOv3 durante o treinamento. Apesar do seu interesse em realizar as alterações a maior parte dos hiperparâmetros em sua forma padrão de utilização foi suficiente para atingir os resultados esperados para o treinamento. HNEWA e RADHA (2021) consideraram também as configurações e hiperparâmetros recomendadas por REDMON e FARHADI (2018) durante o treinamento. Desta maneira, neste trabalho consideraram-se as configurações e hiperparâmetros recomendadas por REDMON e FARHADI (2018).

O hiperparâmetro de *learning rate* (α) define o quão rápido são ajustados os pesos da CNN no treinamento. Se o valor de α é pequeno no início do treinamento o tempo de

treinamento aumenta consideravelmente. Por outro lado, um valor de α grande permite que uma boa solução seja atingida no início. Frequentemente, os valores de α estão entre 0 e 1, então valores próximos de 0 são considerados pequenos enquanto mais próximos de 1 são considerados grandes (BROWNLEE, 2019b). Entretanto, o valor de *loss* da CNN oscilará em torno de um ponto por um tempo consideravelmente longo ou divergirá de forma instável se este valor for mantido. Uma estratégia possível para contornar isso é diminuir o valor de α a partir de um determinado número de iterações, multiplicando-o por um fator (AGGARWAL, 2018). O valor inicial de α adotado foi de 10^{-3} e após 9.600 iterações este valor é multiplicado por 0,1, tornando-se 10^{-4}. Outro hiperparâmetro utilizado para configuração do α foi o de *burn in*. Este hiperparâmetro permite que o valor de α seja aumentado lentamente até um número determinado de iterações. Neste trabalho, considerou-se *burn in* igual a 1000. Estabelecendo que o valor atual de α entre 0 e 1000 iterações assume o valor conforme Equação ((22) (REDMON, 2020):

$$\alpha_{\text{real}} = \alpha_{\text{definido}} \cdot \left(\frac{nº\ de\ iterações}{burn\ in}\right)^4 \tag{22}$$

O aprendizado considerando momento (β) reconhece as atualizações contraditórias da CNN, ou seja, algumas etapas que se anulam e reduzem o efeito das etapas em direção ao caminho correto para atingir o ponto ótimo do ajuste dos pesos. Em função disso, é muito mais adequado realizar as alterações de forma que as atualizações se movam em uma direção "média" considerando as últimas etapas, com isso o denominado "efeito ziguezague" é suavizado. Na Figura 20 se encontra a ilustração da atualização dos valores dos pesos da CNN em busca do ponto ótimo.

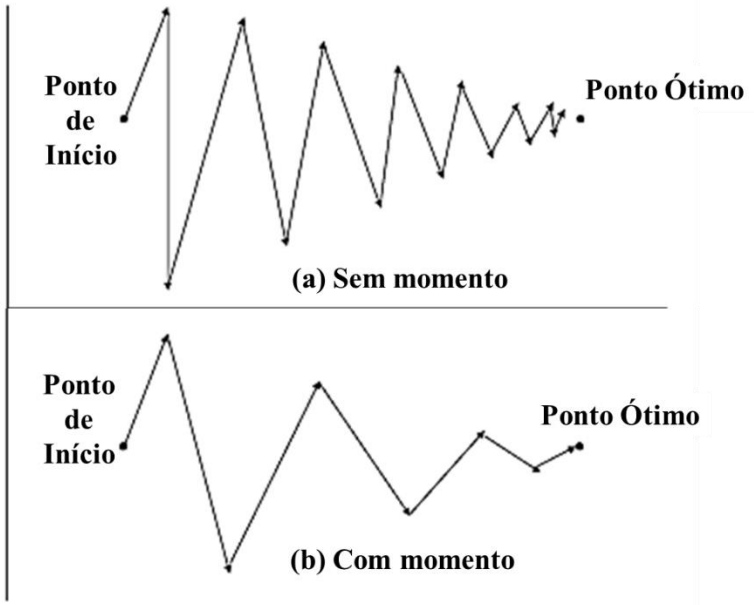

Figura 20 – Comparativo de aprendizado (a) Sem β e (b) com β – Adaptado de (AGGARWAL, 2018)

Na Figura 20 (a), sem a utilização do hiperparâmetro β, é possível identificar o "efeito ziguezague" e as atualizações são realizadas de forma brusca e não se atinge a solução ótima. Enquanto na Figura 20 (b), com a utilização do β, é possível identificar que as atualizações são realizadas de forma mais suave e o ponto ótimo é atingido (AGGARWAL, 2018).

As atualizações para gradiente-descendente (AGGARWAL, 2018) do vetor de parâmetros da CNN (W) são realizados com relação a função *loss* (L) e o valor de α, conforme Equação (23):

$$V \Leftarrow -\alpha \frac{\partial L}{\partial W} \tag{23}$$

com $$W \Leftarrow W + V$$

Em um aprendizado considerando o β o vetor V é modificado considerando-o como um parâmetro de suavização, em que $\beta \in (0,1)$, de acordo com a Equação (24):

$$V \Leftarrow \beta V - \alpha \frac{\partial L}{\partial W}; \;\; W \Leftarrow W + V \tag{24}$$

Além disso, se tem o efeito de mover as atualizações de forma "média" considerando-se valores anteriores. O aprendizado presumindo β é acelerado, porque geralmente aponta para a solução ótima e as oscilações são suavizadas. A ideia básica é de maior preferência as direções consistentes e degraus maiores na direção correta (AGGARWAL, 2018). Para o treinamento, considerou-se o aprendizado com β = 0,9.

O termo época é usado em aprendizado de máquina para indicar o número de vezes que todo o conjunto de dados é utilizado pelo algoritmo. No caso de imagens, é impraticável se utilizar todo o conjunto de uma vez para ser processado, então este é dividido em conjuntos menores. O número de elementos destes conjuntos menores conjunto é conhecido como o hiperparâmetro *batch size*. Eventualmente, mesmo dividindo-se o conjunto de imagens, a unidade de processamento pode não ter capacidade suficiente para lidar com eles. Neste caso, os conjuntos serão adicionalmente subdivididos, o que é configurado pelo hiperparâmetro.

Na Figura 21, ilustra-se que um conjunto de 1000 imagens tem 1 época completa após duas iterações para o processamento de batches com 500 imagens.

Figura 21 – Processamento de uma época a partir de duas iterações.

Neste trabalho aplicaram-se as configurações padrão utilizadas por REDMON e FARHADI (2018), isto é, *batch size* igual a 64 e *subdivisions* igual a 16. A definição do número máximo de iterações (ou *max batches*) é a partir do número de classes levado em conta para detecção pela CNN durante o treinamento multiplicado por 2000, de acordo com a Equação (25):

$$max\ batches = número\ de\ classes \times 2000 \tag{25}$$

Considerou neste trabalho cerca de 70.000 imagens de treinamento do conjunto BDD100K (YU et al., 2020). As classes contempladas foram no total cinco: carros, pedestres, ciclistas/motociclistas, semáforos e sinais de trânsito. Conforme Equação (25) o número máximo de iterações seria igual a 10.000, entretanto para analisar a estabilidade dos resultados após este valor considerou-se o número máximo de iterações igual a 15.000.

Um dos hiperparâmetros avaliado durante o treinamento foi o de altura e largura das imagens de treinamento. Adotaram-se os tamanhos de 416 x 416, 512 x 512, 608 x 608 e 960 x 960 *pixels*. Inicialmente, para o tamanho de imagem de 416 x 416 *pixels* habilitou-se o hiperparâmetro *random*. Este hiperparâmetro redimensiona as imagens de forma aleatória durante as iterações. Depois disso, levou-se em conta o treinamento para o tamanho de imagem de 416 x 416 sem a habilitação de *random* e prosseguiu-se da mesma maneira para o treinamento com demais tamanhos de imagem (512 x 512, 608 x 608 e 960 x 960 *pixels*).

Na Tabela 1, se encontram os hiperparâmetros utilizados para o treinamento da CNN YOLOv3, inicialmente considerando o conjunto de imagens original de treinamento da BDD100K.

Tabela 1 – Hiperparâmetros usados durante o treinamento inicial

Hiperparâmetros	**Valor**
Batch Size	64
Subdivisions	16
(Largura, Altura) pixels	416 x 416 (random = 1), 416 x 416, 512 x 512, 608 x 608, 960 x 960
Learning rate	0,001
nº máximo de iterações	15.000
nº classes	5
burn in	1000

Após realizar a investigação dos diferentes tamanhos de imagens durante o treinamento, os filtros do OpenCV foram implementados. Desta vez, adotando apenas o tamanho de imagem igual a 608 x 608 *pixels*. Os filtros utilizados foram o de erosão, dilatação e o JBF.

A metodologia utilizada para aplicação dos filtros do OpenCV foi de acordo a Figura 22. As imagens para treinamento provieram do depositório BDD100K e os filtros do OpenCV foram aplicados aleatoriamente a elas. Foi realizado um código em *python* para abrir cada uma das imagens do conjunto BDD100K, então sorteado um número aleatório

inteiro, no caso o número 0 ou o número 1. Para isso foi utilizada a função *random.randint()* (PYTHON, 2023). Se o número for 0, a imagem permanece inalterada e é salva. Se o número for 1, então é implementado o filtro do OpenCV na imagem e a mesma é salva. Isto é realizado até que todas as imagens sejam processadas pelo algoritmo.

O código foi desenvolvido na linguagem *python* a partir de notebook da plataforma *Jupyter* (JUPYTER, 2023) e todo o processamento aconteceu em um computador pessoal. Nesta etapa não foram utilizados recursos de GPU ou do Google Colab.

A métrica utilizada para avaliação do treinamento foi *loss*, a qual é comumente utilizada para fins de avaliação de desempenho do treinamento de modelos de detecção de objetos. Para e verificar o impacto da aplicação dos filtros no desempenho da CNN treinada realizaram-se etapas de testes, que serão descritas na próxima seção.

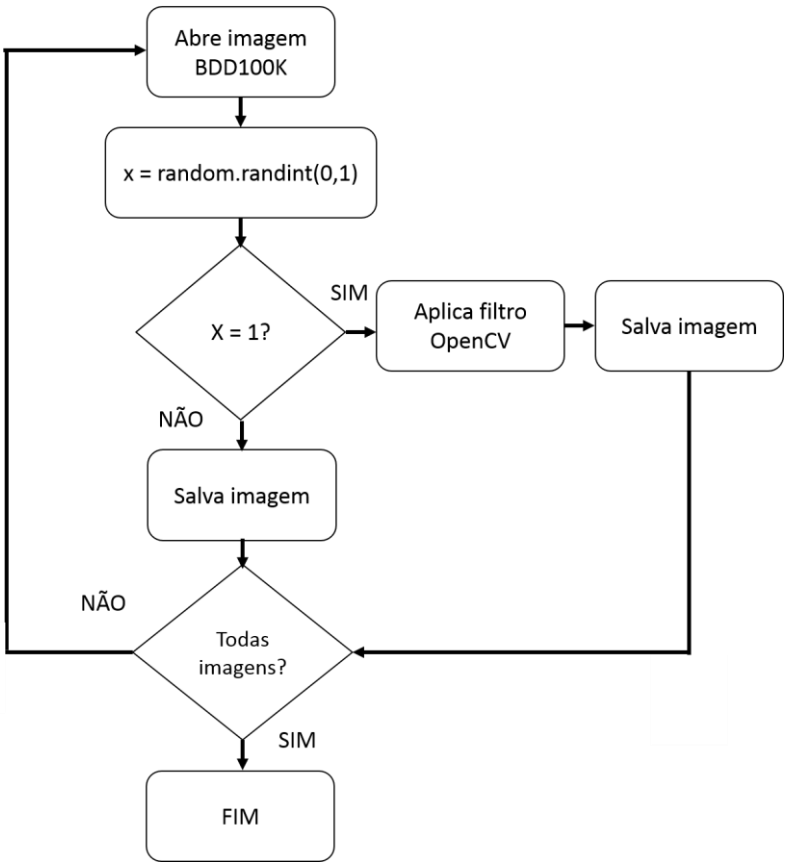

Figura 22 – Aplicação dos filtros durante o treinamento da CNN

7.2 TESTES

Para verificar o desempenho da CNN YOLOv3 realizaram-se testes com os conjuntos de imagens de testes do BDD100K e DAWN, observando que foram tais imagens não foram utilizadas durante o treinamento. Os testes foram realizados na plataforma do Google Colab.

Inicialmente, presumiu-se a avaliação dos tamanhos de imagens para o conjunto BDD100K. Os testes foram realizados com os tamanhos de 416 x 416, 512 x 512, 608 x 608 e 960 x 960 *pixels*. A CNN YOLOv3 foi testada com o respectivo tamanho de imagem levado em conta durante o treinamento. Por exemplo, se a CNN foi treinada com tamanho de 416 x 416 *pixels* o teste foi realizado com tamanho de imagem de 416 x 416 *pixels*. Após isso, realizaram-se testes com diferentes tamanhos de imagem independentemente do tamanho de imagem adotado durante o treinamento. Isto foi realizado com o intuito de avaliar o desempenho da CNN com diferentes tamanhos de imagem não considerados durante o treinamento.

Posteriormente, foram executados os testes com o conjunto DAWN. Desta vez o tamanho de imagem levado em consideração foi somente o de 608 x 608 *pixels*. Pelo fato de as imagens estarem separadas por condição climática (chuva, neve e névoa) procurou-se avaliar o desempenho da CNN para cada condição.

Finalmente, avaliaram-se os testes com a implementação dos filtros do OpenCV em ambos os conjuntos, BDD100K e DAWN. Com o intuito de realizar a comparação dos resultados com e sem a aplicação dos filtros.

Na Figura 23 se encontra o fluxograma utilizado nos testes com a aplicação dos filtros do OpenCV. Para ambos os conjuntos de imagens de testes (BDD100K e DAWN) aplicou-se o filtro do OpenCV de erosão, dilatação e JBG, e o testes com a CNN treinada com a utilização do respectivo filtro durante o treinamento. A aplicação dos filtros durante os testes foi realizada com o intuito de avaliar também as imagens filtradas com o mesmo filtro considerando durante o treinamento. Também, presumiu-se o teste com o conjunto de imagem original para a CNN treinada considerando o filtro do OpenCV. Por fim, a avaliação dos resultados dos testes.

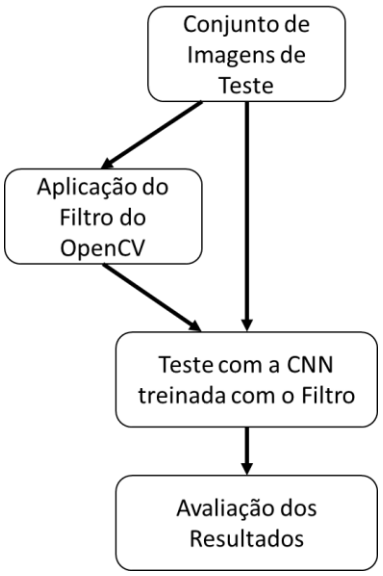

Figura 23 – Fluxograma de testes com a aplicação dos filtros do OpenCV

A única alteração significativa dos hiperparâmetros levados em conta nos testes foram em relação ao *batch size* considerado igual a 1 e *subdivisions* também igual a 1.

8 RESULTADOS

Nesta seção, discutem-se o treinamento e os testes da CNN YOLOv3 para a detecção das classes de carros, sinais de trânsito, semáforos, pessoas e ciclistas/motociclistas. Os resultados foram avaliados de acordo as métricas de avaliação apresentadas na seção 4. A descrição do treinamento se encontra na seção 8.1 e a discussão dos resultados dos testes na seção 8.2.

8.1 TREINAMENTO DA CNN

Para a obtenção dos resultados a serem apresentados, utilizaram-se os recursos da plataforma Google Colab Pro. Além da versão gratuita do Google Colab, existem as opções Google Colab Pro e Pro+, que apresentam recursos adicionais Na Tabela 2, está o comparativo de recursos oferecidos por cada versão do Google Colab. Na primeira linha, se encontram as versões das unidades gráficas de processamento (*Graphic Processing Unit* – GPU) de maior capacidade de processamento, nas versões Pro e Pro+. Na Tabela 3, têm-se as versões de GPUs disponíveis na versão gratuita e nas versões Pro e Pro+, com as respectivas descrições referentes a arquitetura, ano de lançamento, capacidade de memória RAM, possibilidade de expansão da memória RAM no sistema e os preços para o ano de 2022. Para a versão gratuita, na maior parte do tempo, é utilizada a GPU K80, que possui menor capacidade de processamento. Entretanto, mesmo com as versões Pro e Pro+, não é possível selecionar a versão da GPU, pois a seleção é realizada de forma automática.

Todas as versões oferecem duas unidades centrais de processamento virtuais (virtual Central Processing Unit – vCPU), que são unidades físicas associadas ao ambiente do Google Colab. Para a versão gratuita, normalmente é usada a memória RAM (*Random-Access Memory*) de 12 GB, enquanto nas versões Pro e Pro+ é possível utilizar 32 GB e 52 GB para processamento, respectivamente. A versão Pro é oferecida por US$ 9,99/mês e a Pro+ por US$ 49,99/mês.

Tabela 2 – Recursos das versões do Google Colab

Recursos	**Google Colab**	**Google Colab Pro**	**Google Colab Pro+**
GPUs	Predominante K80	K80, P100, T4	P100, T4, V100
CPUs	2 x vCPU	2 x vCPU	2 x vCPU
RAM (GB)	Predominante 12	32	52
Preço (US$)	-	9,99/mês	49,99/mês

Tabela 3 – Comparativo das versões de GPU disponíveis para o Google Colab

GPU	**Arquitetura**	**Ano de Lançamento**	**GPU RAM (GB)**	**CPUs**	**Sistema da RAM (GB)**	**Preço Atual US$ (2022)**
K80	Kepler	2014	12	2 vCPU	13	349
T4	Turing	2018	16	2 vCPU	13 expansível até 25	1.797
P100	Pascal	2016	16	2 vCPU	13 expansível até 25	3.053
V100	Volta	2018	16	2 vCPU	Até 52	3.775

Na Figura 24, tem-se o tempo de treinamento para cada configuração de tamanho de imagem. O treinamento inicial com tamanho de imagem 416 x 416 *pixels* e *random* igual a 1 (416x416r1) aconteceu com a versão gratuita do Google Colab. Para o treinamento com tamanhos de imagem 416 x 416, 608 x 608, 512 x 512 e 960 x 960 *pixels*, utilizou-se a versão do Google Pro.

O tempo de treinamento com tamanho de imagem de 416x416r1 foi o de maior duração, pois os recursos do Google Colab Pro não estavam disponíveis. O menor tempo foi com dimensões de 416 x 416 *pixels*, devido à utilização dos recursos do Google Colab Pro e na maior parte do tempo do treinamento foi utilizada a GPU P100. Para 512 x 512 e 608 x 608 *pixels* foram alternadas as GPUs P100 e T4 e para 960 x 960 pixels foi utilizada, principalmente, a GPU T4. Este é o motivo que a duração foi maior para 960 x 960 *pixels* do que para os demais tamanhos de imagem com recursos do Google Colab Pro.

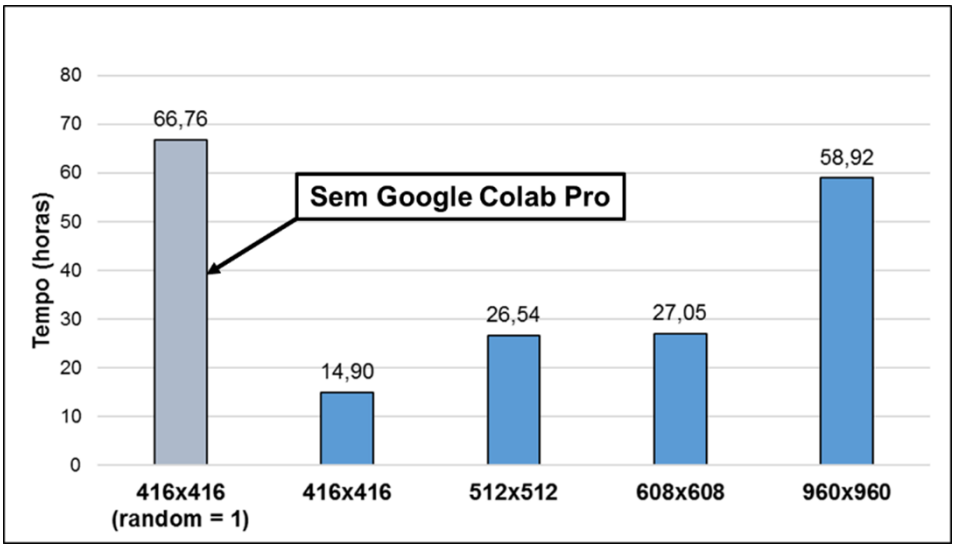

Figura 24 – Tempo de treinamento da CNN

De forma geral, os recursos do Google Colab Pro levaram a um menor tempo de treinamento, não somente pela utilização de GPUs mais potentes, mas também por permitir um tempo de inatividade de até 24 horas. Na versão gratuita do Google Colab, o tempo de inatividade é de 12 horas e a verificação de atividade é realizada cerca de 3 horas depois do início do treinamento. Caso identifique-se que não há nenhuma atividade do usuário, o ambiente é desconectado. O treinamento da CNN dentro das 3 horas não é perdido, porém todo o ambiente precisa ser reinicializado, aumentando ainda mais o tempo de treinamento, o que não foi contabilizado na Figura 24. Com o Google Colab Pro, as verificações na máquina ocorrem a cada 6 horas.

Como a seleção da GPU é realizada de forma automática, não é possível estabelecer uma correlação direta de tempo de duração de treinamento e o tamanho de imagem. A utilização do Google Colab Pro com as GPUs V100 e P100 pode diminuir respectivamente, em média de 146% e 63% o tempo de treinamento quando comparada ao tempo com a GPU T4 (LI, 2022).

Na Figura 25, apresentam-se os resultados de treinamento de *Loss versus* o número de iterações para imagens de diferentes tamanhos: 960 x 960, 608 x 608, 416 x 416 *pixels*, com parâmetro *random* igual a 0 (*random* = 0), e 416 x 416 *pixels* com o hiperparâmetro *random* igual a 1 (416x416r1). Apresenta-se a evolução do parâmetro *Loss* em função do número de iterações de acordo com o hiperparâmetro de *learning rate*. Na região (i) se encontram as primeiras 250 iterações de treinamento, onde é possível observar que o valor

de *Loss* é elevado, isto porque, inicialmente, os pesos da CNN não estão ajustados para aquele conjunto de imagens e os erros relacionados à classificação do objeto, à sua posição, são altos. Com o valor de *learning rate* sendo ajustado de forma mais agressiva neste período o valor de *Loss* é diminuído. Após as 250 iterações, na região (ii), é possível observar que ocorre uma diminuição do valor de *Loss* e ocorre certa estabilização após 1000 iterações, momento em que o valor de *learning rate* fica igual a 0,001. No caso deste trabalho, considerou-se o valor de *learning rate* igual a 0,001 a partir de 1000 iterações (REDMON; FARHADI, 2018). Para isso, configurou-se o hiperparâmetro *burn_in* igual a 1000 e, entre 0 e 1000 iterações, o valor *learning rate* é atualizado de 0 a 0,001. A partir de 1000 iterações, região (iii), o valor de *learning rate* permanece igual a 0,001 e somente após 9.600 iterações o valor de learning rate é atualizado para 0,0001. Entretanto, na região (iii) há tendência de estabilidade do valor de *Loss*.

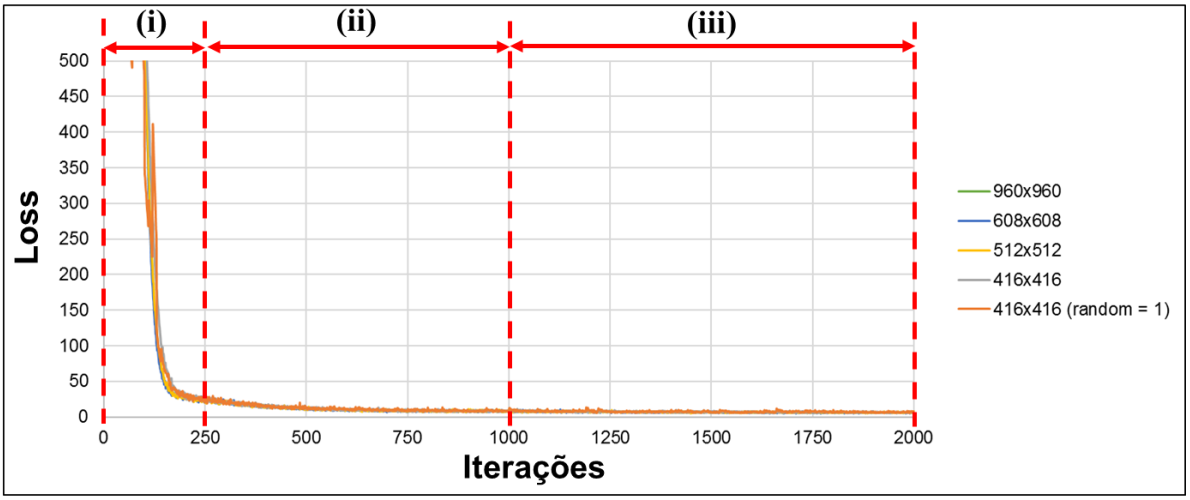

Figura 25 – *Loss* durante o treinamento para as 2000 primeiras iterações

Utilizando-se uma média móvel de 200 iterações, os resultados de *Loss versus* iterações para o treinamento da CNN com 15.000 iterações se encontra na Figura 26. Identifica-se que o valor de *Loss* diminui nas primeiras iterações e, após 1000 iterações, há tendência de estabilidade.

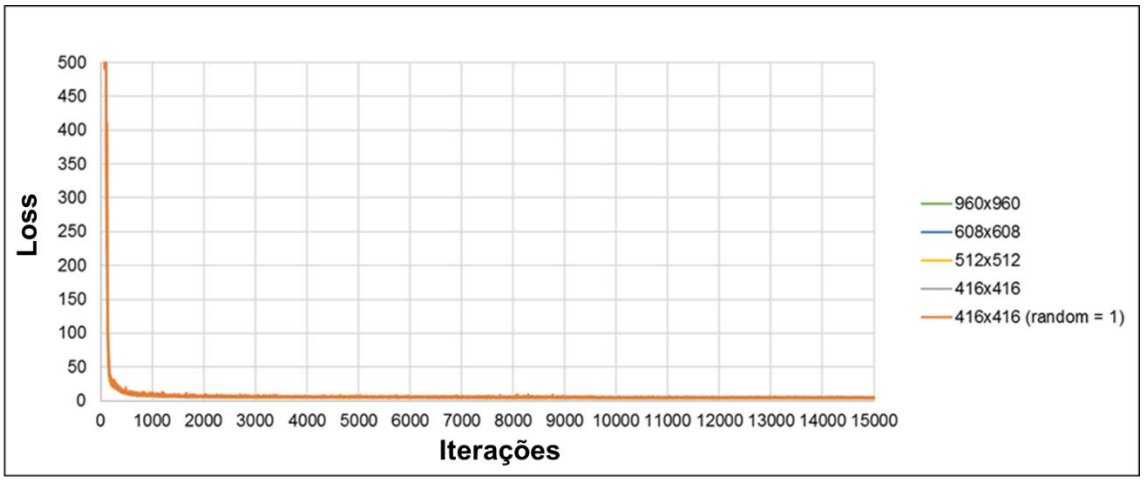

Figura 26 – *Loss* durante o treinamento para 15.000 iterações

Na Figura 27, mostra-se ampliada a região partir de 4000 iterações com uma média móvel de 200 iterações. É possível notar que entre 4000 e 10.000 iterações há tendência de diminuição de *Loss* e a partir de 10.000 iterações os valores se estabilizam, para qualquer tamanho de imagem usado no treinamento. Da Figura 27, torna-se claro que não há diferenças significativas durante o treinamento da CNN para imagens de tamanhos distintos.

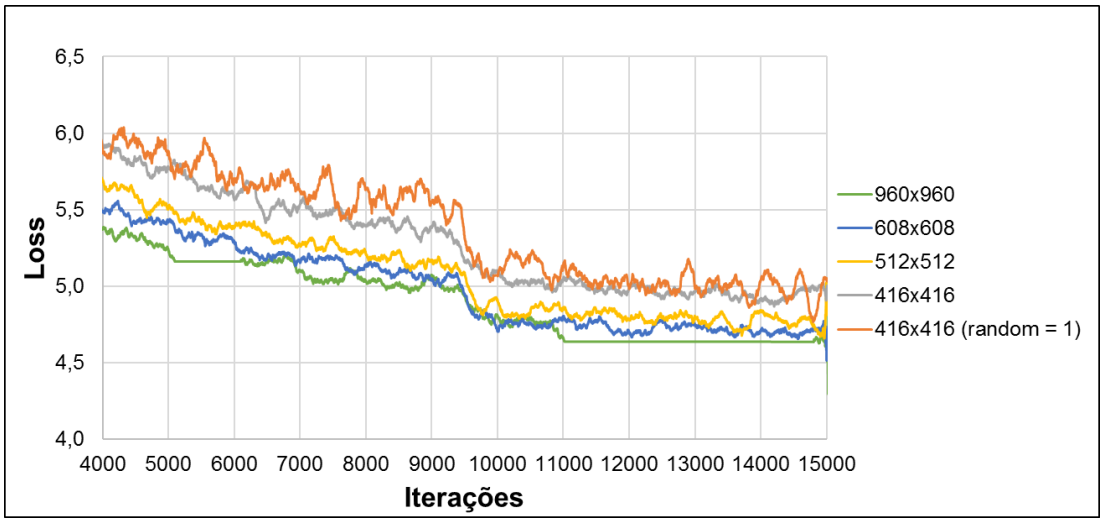

Figura 27 – *Loss* ampliado a partir de 4000 iterações

8.2 TESTES

Após o treinamento da CNN, testou-se o seu desempenho. Inicialmente, com imagens que pertenciam ao mesmo depositório BDD100K (YU et al., 2020), mas que não foram utilizadas durante o treinamento.

Posteriormente, testou-se a CNN com o conjunto de imagens DAWN (KENK; HASSABALLAH, 2020). As imagens consideradas nesse caso também não foram utilizadas durante o treinamento.

Por fim, após os testes iniciais com os depositórios BDD100K e DAWN testou-se a CNN treinada com os filtros do OpenCV tanto com imagens na condição original como com filtros do OpenCV.

8.2.1 Conjunto de Imagens BDD100K

Durante o treinamento da CNN considerou-se imagens com tamanho de: 416 x 416, 512 x 512, 608 x 608 e 960 x 960 *pixels*. Para os testes com imagens do depositório BDD100K, utilizaram-se imagens com tamanho de 416 x 416, 512 x 512, 608 x 608 e 960 x 960 *pixels*. Os testes foram realizados com imagens de mesmo tamanho considerado durante o treinamento.

Na Figura 28 se encontram o número de instâncias das classes de carros, sinais de trânsito, semáforos, pessoas e ciclistas/motociclistas do conjunto de imagens de treinamento do BDD100K. Nota-se que a classe de carros é a que possui maior número de instâncias, em sequência as classes de sinais de trânsito, semáforos, pessoas e a que possui menor número é a de ciclistas/motociclistas.

Os resultados de Precisão Média (*Average Precision* - AP) se encontram na Figura 29 para cada classe e tamanho de imagem, apresentam-se os resultados dos testes avaliados pela AP com as classes ordenadas de acordo com o desempenho. Para a classe de carros obteve-se o melhor desempenho pelo fato de ocuparem regiões maiores nas imagens e, assim, possuírem áreas maiores para a extração de características quando comparados aos objetos das demais classes. Nota-se que, à medida que aumenta o tamanho das imagens, o desempenho da CNN também aumenta, repetindo-se o mesmo comportamento para demais classes. O segundo melhor desempenho foi para os sinais de trânsito, pois são feitos de materiais com alta reflexão o que facilita a detecção pela CNN. Enquanto a classe de semáforos e pessoas apresentam o terceiro e quarto melhor desempenho, respectivamente. A detecção de objetos das classes de semáforo é desafiadora por envolver áreas relativas pequenas (HNEWA; RADHA, 2021), o mesmo ocorre com a detecção de pessoas. Conforme Figura 28, a classe com maior número de instâncias é a classe de carros, o que é um fator determinante para que esta classe tenha os melhores resultados de AP. Por outro

lado, a classe de ciclistas/motociclistas é a classe com o menor número de instâncias e com a menor AP.

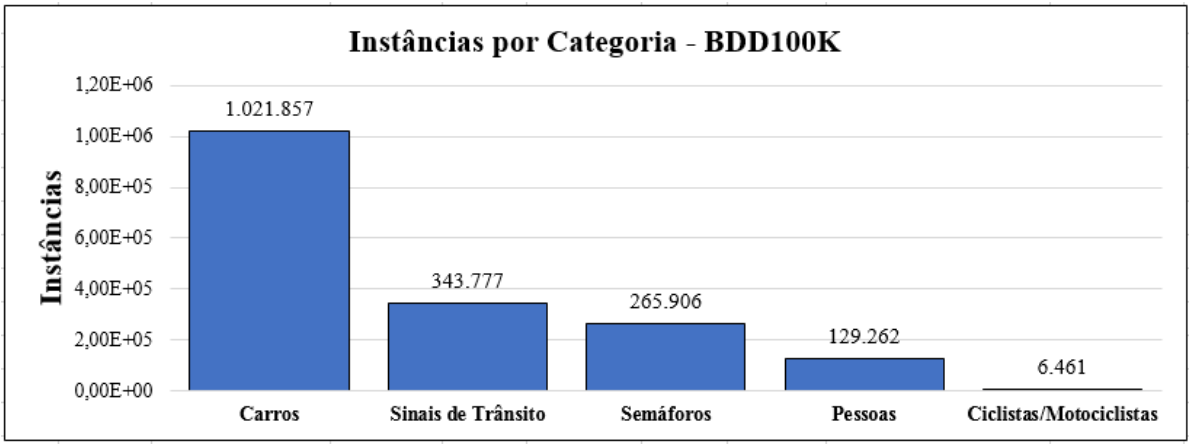

Figura 28 – Instâncias das classes utilizadas do depositório BDD100K (YU et al., 2020)

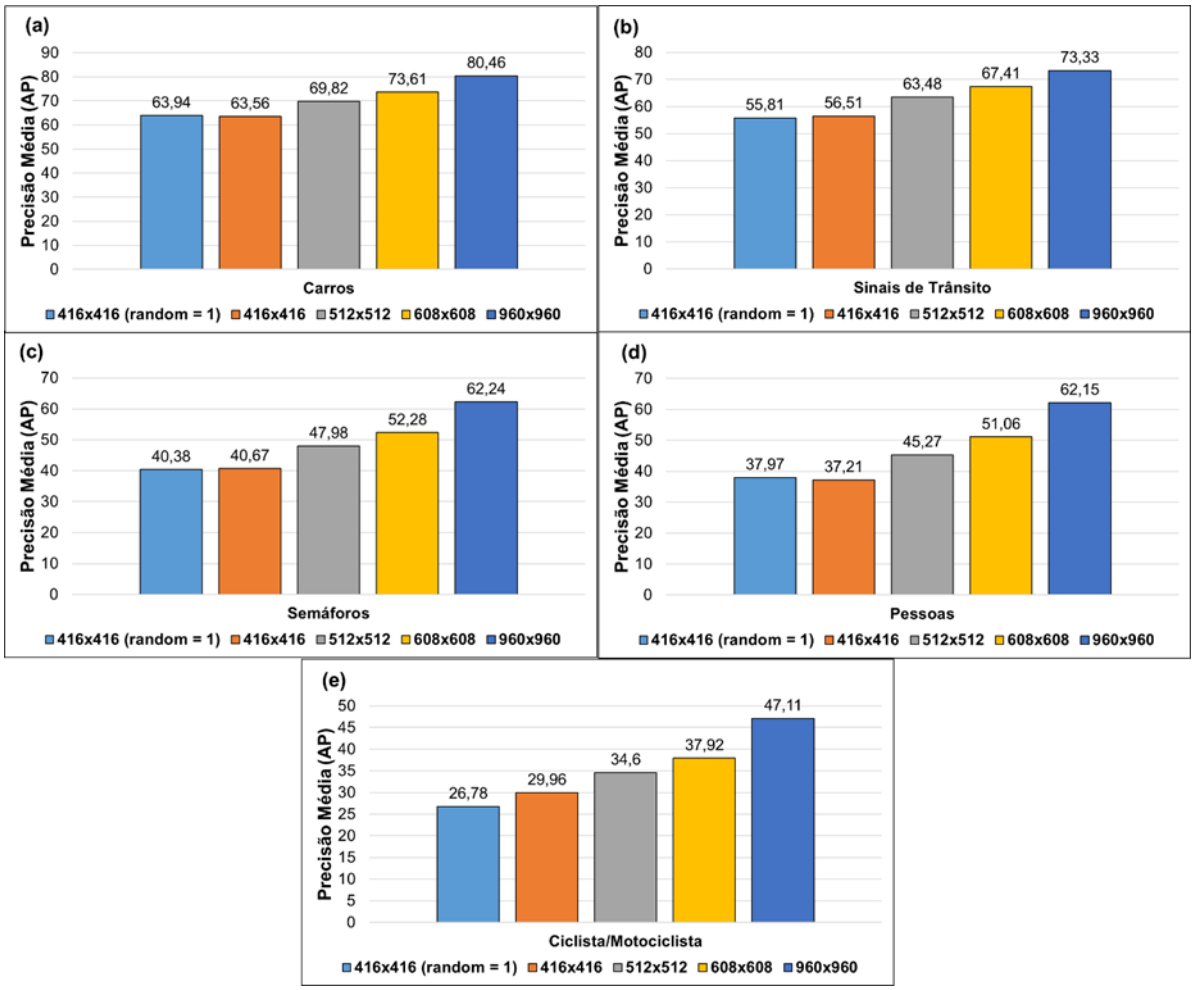

Figura 29 – Precisão média para o conjunto de imagens BDD100K para as classes de (a) Carros, (b) Sinais de Trânsito, (c) Semáforos, (d) Pessoas e (e) Ciclista/Motociclista

Verificando-se os resultados de mAP, na Figura 30, observa-se um aumento de cerca de 20% na métrica de mAP à medida que aumenta o tamanho da imagem de entrada, comparando-se os resultados com 960 x 960 *pixels* em relação aos resultados com 416 x 416 *pixels*.

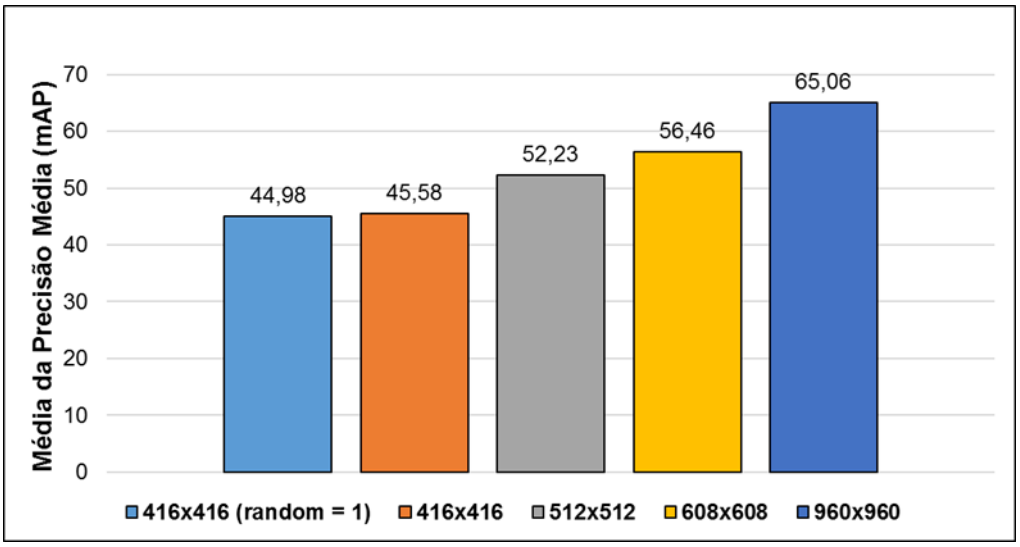

Figura 30 – mAP para com o conjunto BDD100K

Os resultados da Medida-F, na Figura 31, guardam certa semelhança com os de mAP, com o melhor resultado para 960 x 960 *pixels*. Conforme visto anteriormente, aumentando o tamanho de imagem para treinamento, obtém-se melhor desempenho, pois mais características podem ser extraídas da imagem. Entretanto, isso leva a um maior tempo de processamento e caberá ao usuário escolher a relação custo-benefício neste processo.

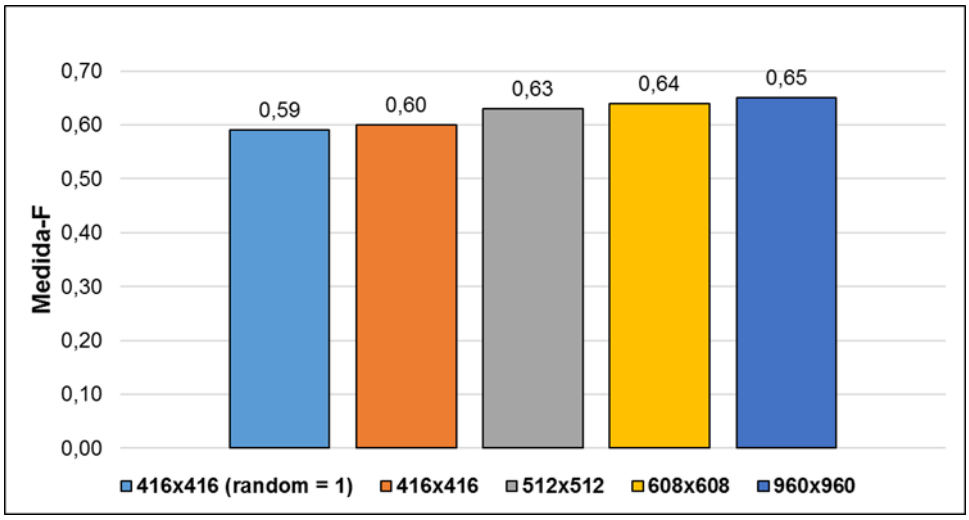

Figura 31 – Medida-F para o conjunto BDD100K

Nas Figura 32 e Figura 33, se encontram os resultados de mAP e Medida-F, respectivamente, para os testes com tamanhos de imagem de 416 x 416, 512 x 512, 608 x 608 e 960 x 960 *pixels* para CNNs treinadas com tamanhos de imagem de 416x416r1, 416 x 416, 512 x 512, 608 x 608 e 960 x 960 *pixels*.

Observando-se a Figura 32 identifica-se que conforme aumenta o tamanho de imagem de teste, o desempenho aumenta para as CNNs treinadas com 416x416r1, 512 x 512, 608 x 608 e 960 x 960 *pixels*. Somente a CNN treinada com 416 x 416 *pixels* apresentou queda no desempenho para os testes com 960 x 960 pixels. Isso se deve ao fato do tamanho da imagem de teste com 960 x 960 *pixels* ser quase o dobro do tamanho das imagens utilizadas para o treinamento, prejudicando a extração de características pela CNN e as predições. Isso não se observa nos resultados com 416x416r1, pois apesar do tamanho da imagem considerado para treinamento ser de 416 x 416 *pixels* quando se considera o hiperparâmetro *random = 1* outros tamanhos de imagem são considerados durante o treinamento.

O mesmo comportamento dos resultados de mAP se repetem para a medida-F, de acordo com a Figura 33. Observa-se que os melhores resultados foram para os testes com tamanho de imagem igual ao utilizado durante o treinamento e o melhor desempenho foi para a CNN treinada com 960 x 960 *pixels* e testada com tamanho de imagem de 960 x 960 *pixels*.

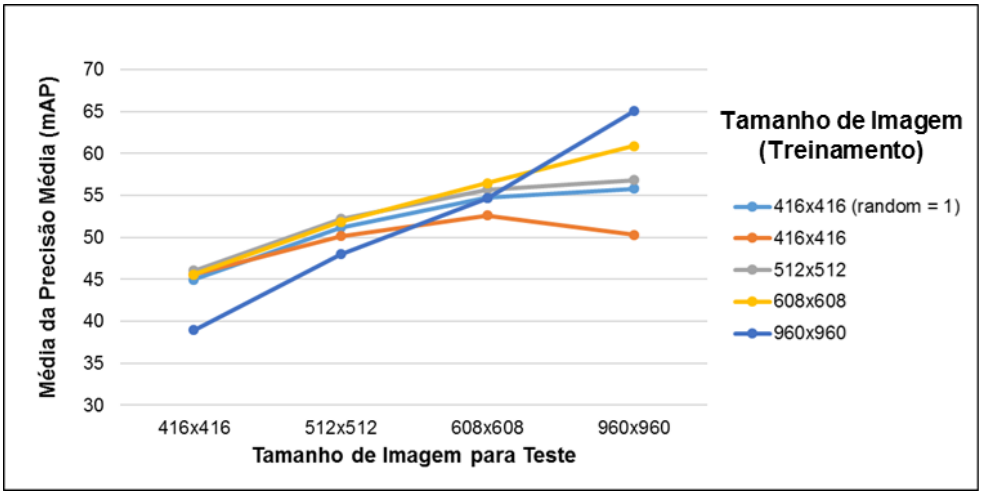

Figura 32 – mAP para diferentes tamanhos de imagem de entrada

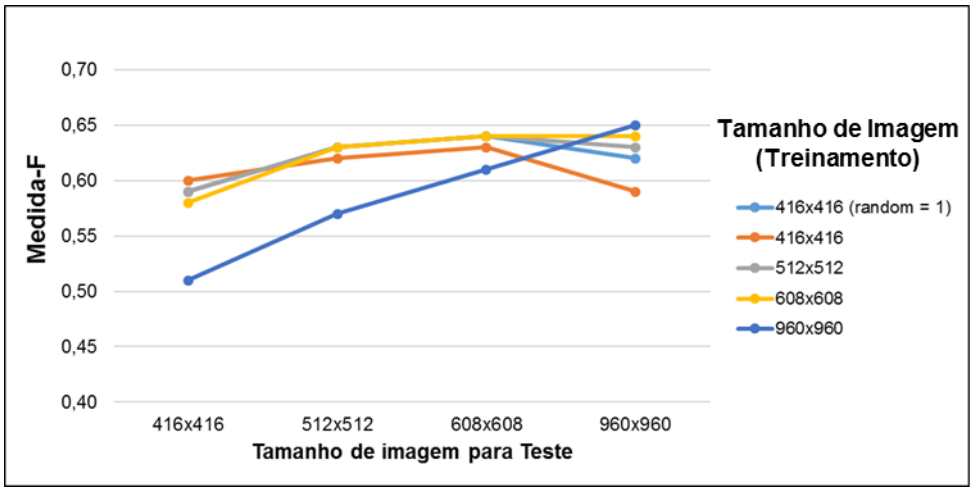

Figura 33 – Medida-F para diferentes tamanhos de imagem de entrada

Avaliou-se o tempo de processamento com uma GPU T4 como função do tamanho da imagem de teste do conjunto do BDD100K e os resultados aparecem na Figura 34. Observa-se que o tempo de processamento aumenta à medida que se aumenta o tamanho da imagem. No entanto, o teste é realizado em um ambiente colaborativo com GPU e em um sistema real automotivo o tempo de processamento deve ser considerado em conjunto com a dinâmica veicular, o que foge ao escopo deste trabalho.

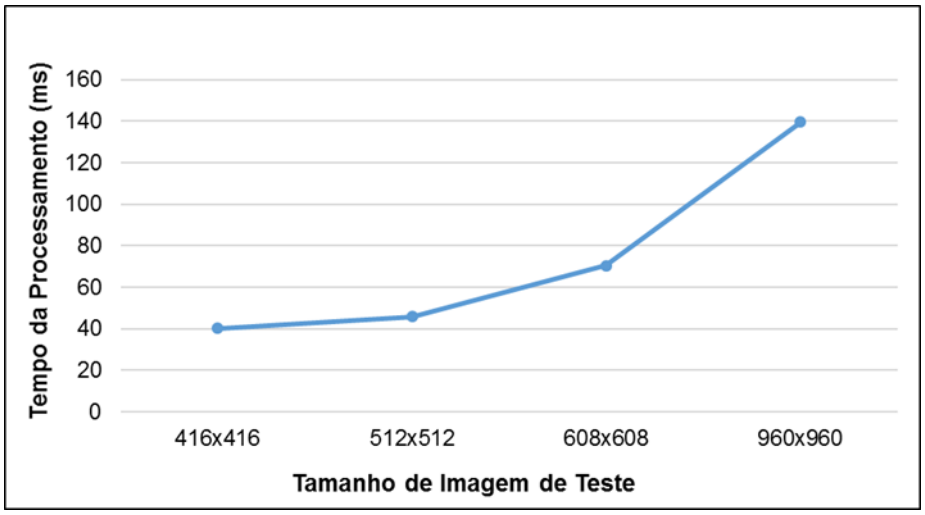

Figura 34 – Tempo de processamento para diferentes tamanhos de imagem

8.2.2 Conjunto de Imagens DAWN

Após realizar os testes com o conjunto BDD100K, realizaram-se os testes com a CNN com o depositório de imagens DAWN. Para estes testes considerou-se a CNN treinada com imagens de tamanho 608 x 608 *pixels* e os testes também foram realizados

com imagem de tamanho 608 x 608 pixels. Os resultados de AP para a classe de Carros se encontram na Figura 35.

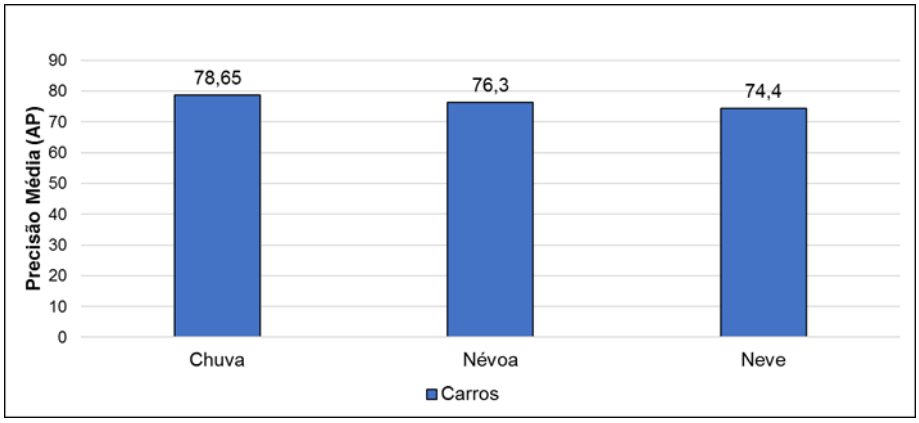

Figura 35 – Testes com o conjunto de imagens DAWN – Classe de Carros

Nota-se que, neste conjunto de imagens testado, os melhores resultados foram na condição de chuva, em seguida na condição de névoa e neve. A neve é uma das condições climáticas mais desafiadoras para detecção de objetos. Nesta condição os objetos são parcialmente ou totalmente cobertos, o que provoca perdas de características importantes dos veículos (DING et al., 2022). Na Figura 36 se encontram imagens utilizadas durante os testes na condição de neve, onde é possível identificar cobertura parcial ou total dos veículos. Por este motivo na condição de neve a CNN apresentou menor desempenho para esta classe.

Figura 36 – Imagens de Carros na condição Neve do conjunto de imagens BDD100K

Na Figura 37 se encontra a distribuição do número de instâncias da classe de Carros. Identifica-se que o número de instâncias varia de acordo com a condição climática

e apesar de o número de imagens ser similar entre as condições o número de instâncias varia.

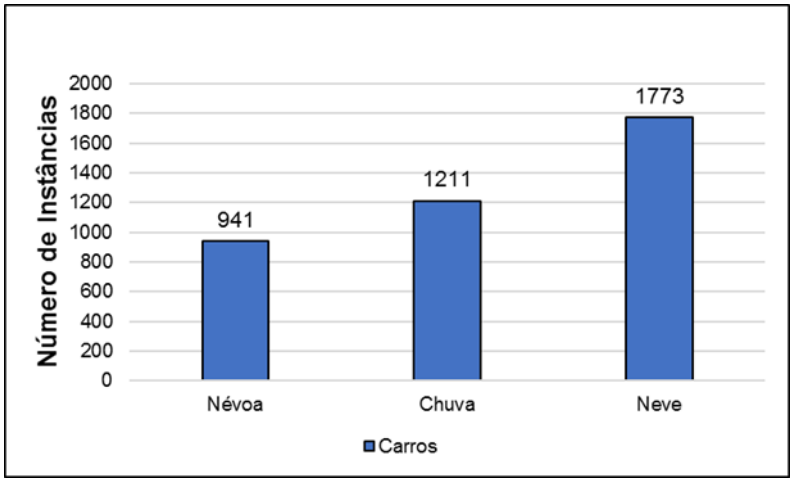

Figura 37 – Distribuição das instancias de Carros para o conjunto de imagens DAWN

Na Figura 38 se encontram os resultados de AP para a classe de Pessoas. Verifica-se que os melhores resultados foram para a condição de chuva, em seguida para a condição de névoa e condição de neve.

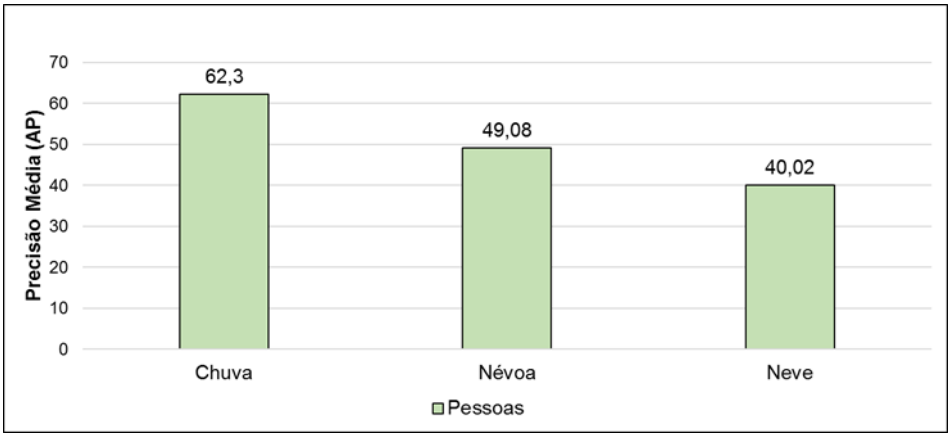

Figura 38 – Testes com o conjunto de imagens DAWN – Classe de Pessoas

Como citado anteriormente a classe de Pessoas é uma das classes desafiadoras para detecção pelas CNNs. Como pode ser verificado na Figura 39, as pessoas podem estar localizadas em ângulos e posições diferentes, além de ocuparem uma região menor ou maior na imagem. Adicionalmente, neste conjunto de imagens se tem a influência da condição climática o que altera as vestimentas e regiões que possuem características importantes podem ser cobertas. Novamente identifica-se que o pior desempenho da CNN é na condição com neve.

Figura 39 – Imagens da classe de Pessoas do conjunto DAWN

Na Figura 40 se encontra a distribuição do número de instâncias da classe de Pessoas. Novamente identifica-se que a distribuição das instâncias não é uniforme.

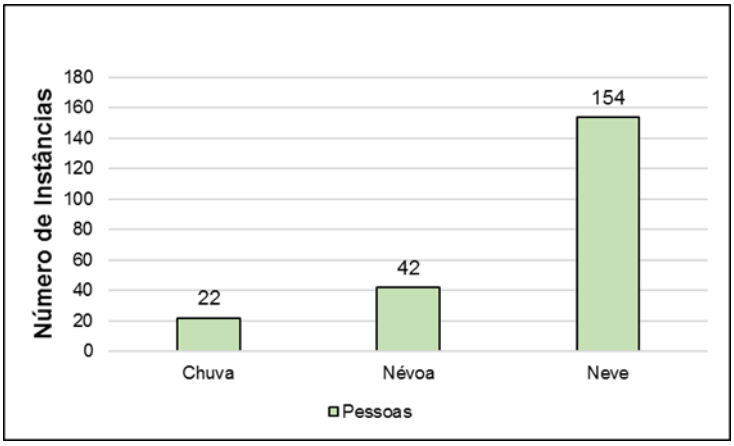

Figura 40 – Distribuição das instâncias de Pessoas para o conjunto DAWN

Observa-se que a condição adversa que mais impacta no desempenho da detecção de objetos é a de neve, seguida por névoa e chuva. A classe de carros apresentou melhor resultado quando comparado a classe de pessoas, isto se deve ao fato dos carros ocuparem maiores regiões e as pessoas pertencerem a uma classe desafiadora para detecção de objetos.

8.2.3 Conjunto de Imagens BDD100K – com filtros do OpenCV

Após realizar os testes iniciais com o conjunto BDD100K, realizaram-se os testes adicionais com a CNN desta vez utilizando-se os filtros do OpenCV. Similarmente como nos demais testes a CNN foi treinada com imagens de tamanho 608 x 608 *pixels* e os testes também realizados com imagem de tamanho 608 x 608 *pixels*.

A identificação do conjunto de imagens de testes do BDD100K se deu pelo nome 'val'. Para testar a CNN treinada com o filtro de Erosão considerou-se o conjunto de imagens 'val' e também o mesmo conjunto, porém com todas as imagens com o filtro de Erosão aplicado identificado como 'val_A'. Da mesma maneira a CNN treinada com o filtro de Dilatação foi testada com o conjunto de imagens 'val' e com o conjunto 'val' com o filtro de Dilatação aplicado, denominado como 'val_B'. Por fim, a CNN treinada com o filtro *Joint Bilateral Filter* (JBF) foi testada com o conjunto 'val' e com o conjunto 'val' com o filtro JBF aplicado (val_C).

A divisão dos conjuntos de imagens de testes se encontra na Tabela 4. Na primeira coluna se encontra o nome considerado do conjunto de imagens. Enquanto na segunda coluna se encontra a descrição do conjunto. Observando-se que todos são variações do conjunto BDD100K.

Tabela 4 – Variações do conjunto de imagens de testes do BDD100K

Conjunto de Imagens	**Descrição**
val	Imagens de testes do conjunto BDD100K
val_A	Imagens de testes do conjunto BDD100K com a aplicação do filtro de Erosão
val_B	Imagens de testes do conjunto BDD100K com a aplicação do filtro de Dilatação
val_C	Imagens de testes do conjunto BDD100K com a aplicação do filtro JBF

Na Figura 41, se encontram os resultados de Precisão Média (*Average Precision* - AP) para cada classe de objeto. Primeiramente, identificado como 'val' se refere a CNN treinada com o conjunto de imagens originais do BDD100K e testada com o conjunto de

testes ‘val’. Na sequência, os testes realizados com os conjuntos ‘val’ e ‘val_A’, porém desta vez com a CNN treinada com o BDD100K considerando o filtro de Erosão. Em seguida, se encontram os testes realizados com os conjuntos ‘val’ e ‘val_B’, considerando a CNN treinada com o BDD100K e levando em conta o filtro de Dilatação. Finalmente, apresentam-se os testes realizados com os conjuntos ‘val’ e ‘val_C’ com a CNN treinada com o BDD100K e considerando o filtro JBF durante o treinamento.

Os melhores resultados são para a classe de Carros, seguido por Sinais de Trânsito, Semáforos, Pessoas e Ciclistas/Motociclistas. Identifica-se que para todas as classes testadas no conjunto ‘val’ a aplicação dos filtros de Erosão, Dilatação e JBF não contribuíram para o aumento do desempenho da CNN quando se compara o desempenho da CNN sem a aplicação de filtros.

Na Figura 42 apresenta-se a métrica de mAP para os testes realizados com as variações do conjunto ‘val’ e considerando a CNN treinada com o conjunto BDD100K assim como levando em conta os filtros de Erosão, Dilatação e JBF durante o treinamento. Enquanto na Figura 43 se encontram os resultados dos mesmos testes, porém é mostrada a Medida-F. É possível observar o mesmo comportamento, em que a aplicação dos filtros não contribuiu para o aumento do desempenho da CNN nos testes com o conjunto ‘val’.

O que leva a concluir que a aplicação dos filtros do OpenCV de Erosão, Dilatação e JBF não aumentaram o desempenho da CNN na detecção de objetos nos testes do conjunto de imagens do BDD100K.

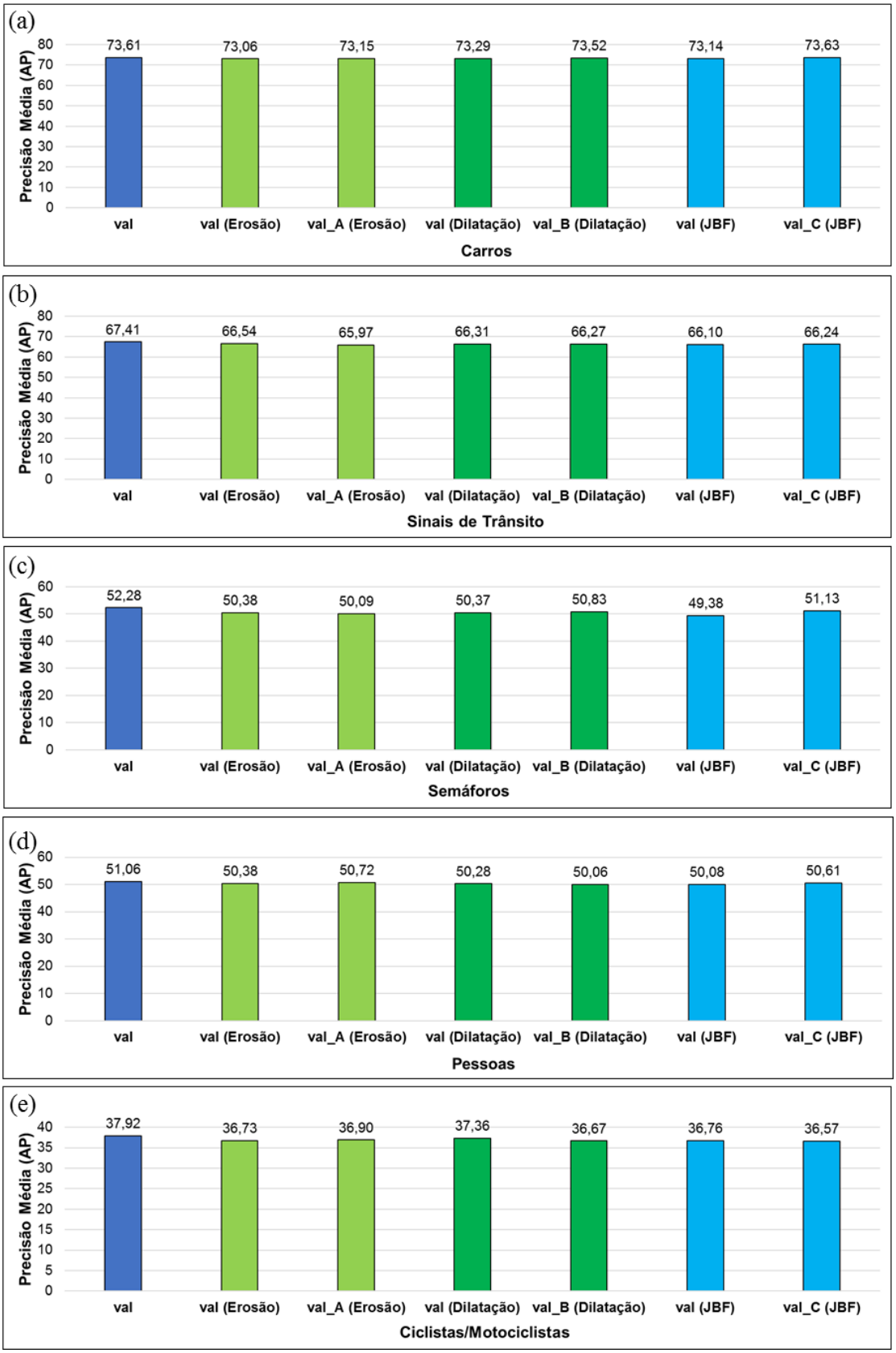

Figura 41 – Precisão média para o conjunto de imagens BDD100K para as classes de (a) Carros, (b) Sinais de Trânsito, (c) Semáforos, (d) Pessoas e (e) Ciclista/Motociclista

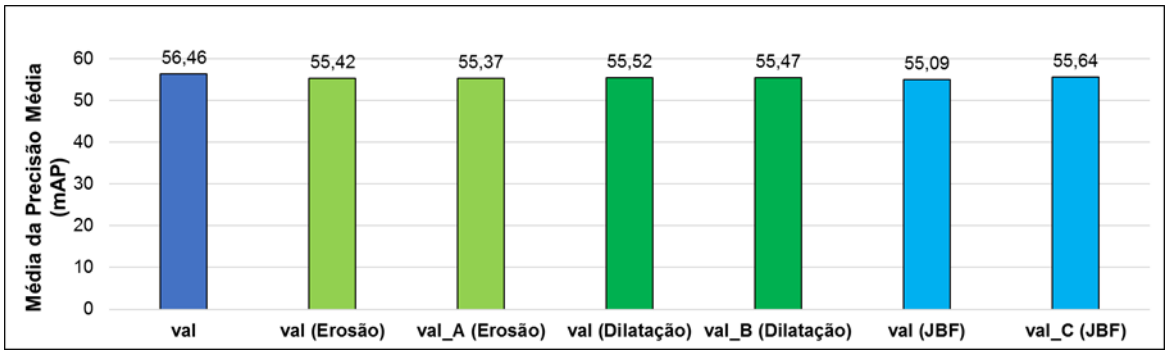

Figura 42 – Resultados de mAP para o com o conjunto de imagens BDD100K

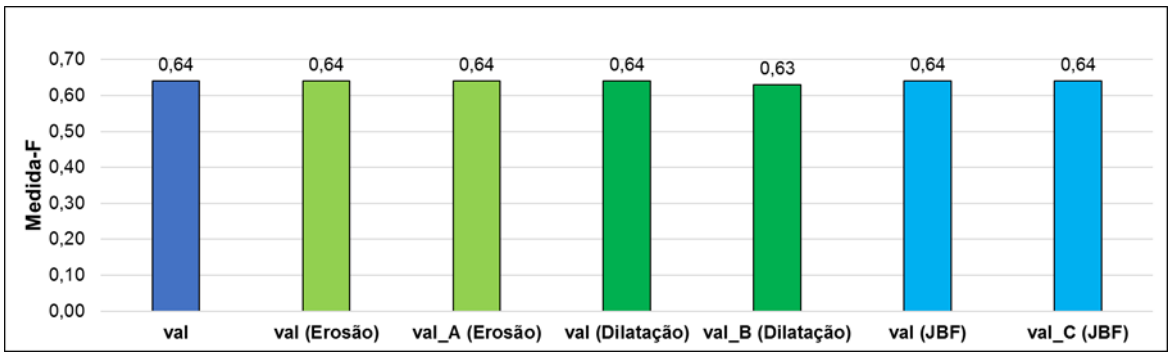

Figura 43 – Resultados da Medida-F para o com o conjunto de imagens BDD100K

8.2.4 Conjunto de Imagens DAWN – com filtros do OpenCV

Na sequência, testou-se a CNN treinada com o OpenCV utilizando o depositório DAWN, que possui imagens com neve, névoa e chuva. Da mesma maneira, a CNN foi treinada com imagens de tamanho 608 x 608 *pixels* e o mesmo tamanho de imagem usado para os testes, que foram separados por condição climática.

As imagens em condição de neve do depositório são identificadas por ‘Neve’ e o primeiro resultado representa o teste com o conjunto com a CNN treinada sem a aplicação de filtros. O filtro de Erosão foi utilizado para o treinamento da CNN e os testes foram realizados para o conjunto de ‘Neve’ e o conjunto ‘Neve’ com todas as imagens com o filtro de Erosão aplicado, identificado como ‘Neve_A’. Similarmente, a CNN treinada com o filtro de Dilatação foi testada com o conjunto de imagens ‘Neve’ e com o conjunto ‘Neve’ com o filtro de Dilatação aplicado, identificado como ‘Neve_B’. Por fim, a CNN treinada com o filtro *Joint Bilateral Filter* (JBF) foi testada com o conjunto ‘Neve’ e com o conjunto ‘Neve’ com o filtro JBF aplicado (Neve_C).

Na Figura 44 (a) estão os resultados de AP para a classe de Carros e na Figura 44 (b) os resultados de AP para as classes de Pessoas. Para a classe de Carros, observa-se que a aplicação dos filtros de Erosão e Dilatação aumentou a métrica de AP, enquanto a aplicação do filtro JBF diminuiu o desempenho da CNN. Para a classe de Pessoas, identifica-se que somente o conjunto de imagens de Neve com a aplicação dos filtros de Erosão obteve resultado superior à condição sem utilização de filtros. Os demais resultados com os filtros de Dilatação e JBF resultaram em menor desempenho da CNN.

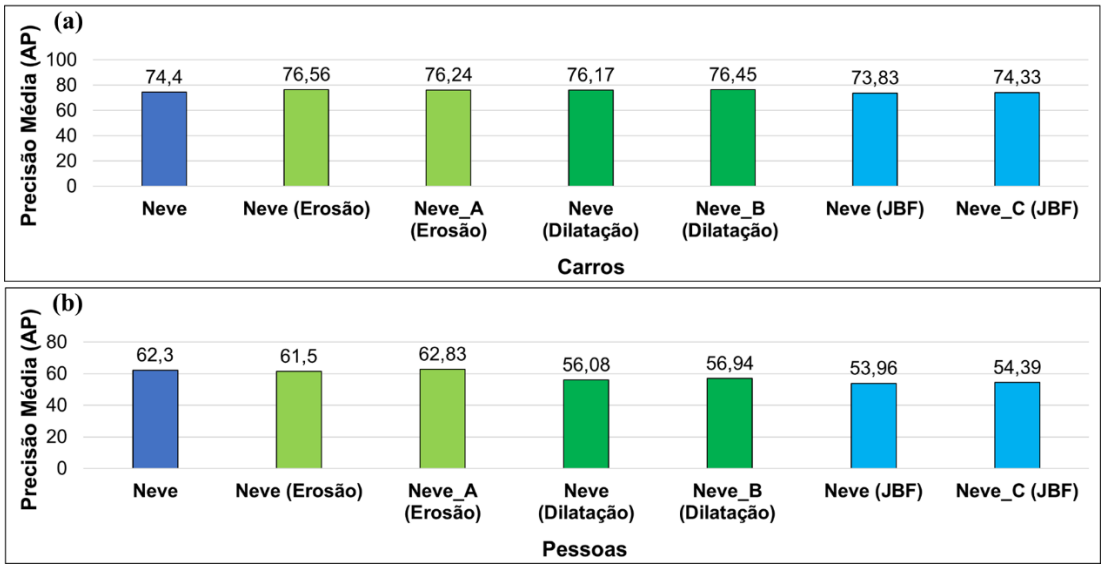

Figura 44 – Precisão média para o conjunto de imagens com neve do DAWN para as classes de (a) Carros e (b) Pessoas

Na Figura 45, encontram-se os resultados de mAP para o conjunto de imagens com neve do DAWN. Nota-se que os melhores resultados da CNN foram obtidos utilizando o filtro de Erosão durante o treinamento e testando o conjunto original de ‘Neve’ e o conjunto com as imagens com o filtro aplicado, ‘Neve_A’. Os demais resultados mostram que os filtros de Dilatação e JBF não aumentaram o desempenho da CNN para o conjunto ’Neve’, observando-se a métrica de mAP.

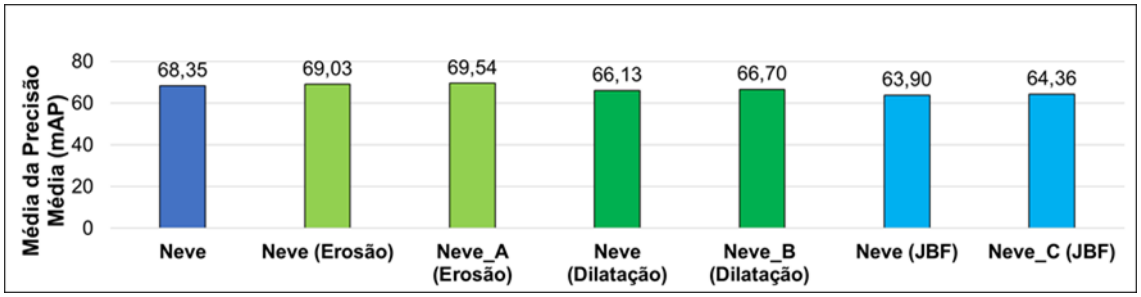

Figura 45 – mAP para o conjunto de imagens com neve do DAWN

Na Figura 46 (a) e (b) o primeiro resultado se refere ao teste com o depositório de imagens com névoa (“Névoa”) com a CNN treinada sem a aplicação de filtros. O filtro de Erosão foi utilizado para o treinamento da CNN e os testes foram realizados considerando o conjunto de ‘Névoa’ e todas as imagens com o filtro de Erosão aplicado, identificado como ‘Névoa_A’. Da mesma maneira, a CNN treinada com o filtro de Dilatação foi testada com o conjunto de imagens ‘Névoa’ e com o conjunto ‘Névoa’ com o filtro de Dilatação aplicado, identificado como ‘Névoa_B’. Por fim, a CNN treinada com o filtro JBF foi testada com o conjunto ‘Névoa’ e com o conjunto ‘Névoa’ com o filtro JBF aplicado (Névoa_C).

Na Figura 46 (a) apresenta-se a métrica de AP para a classe de Carros e na Figura 46 (b) para as classes de Pessoas. Para a classe de Carros, nota-se que os filtros de Erosão e Dilatação aumentaram o desempenho da CNN, enquanto o filtro JBF o diminuiu. Para a classe de Pessoas, identifica-se que a aplicação dos filtros aumentou a métrica de AP, com os melhores resultados com a aplicação dos filtros de Erosão, Dilatação e JBF, respectivamente.

Na Figura 47, estão os resultados de mAP para o conjunto de imagens com névoa do DAWN. Observa-se que a aplicação dos filtros para a condição de névoa melhorou os resultados em relação à condição sem filtro, notando-se que o filtro de Erosão apresentou os melhores resultados seguido pelos filtros de Dilatação e JBF.

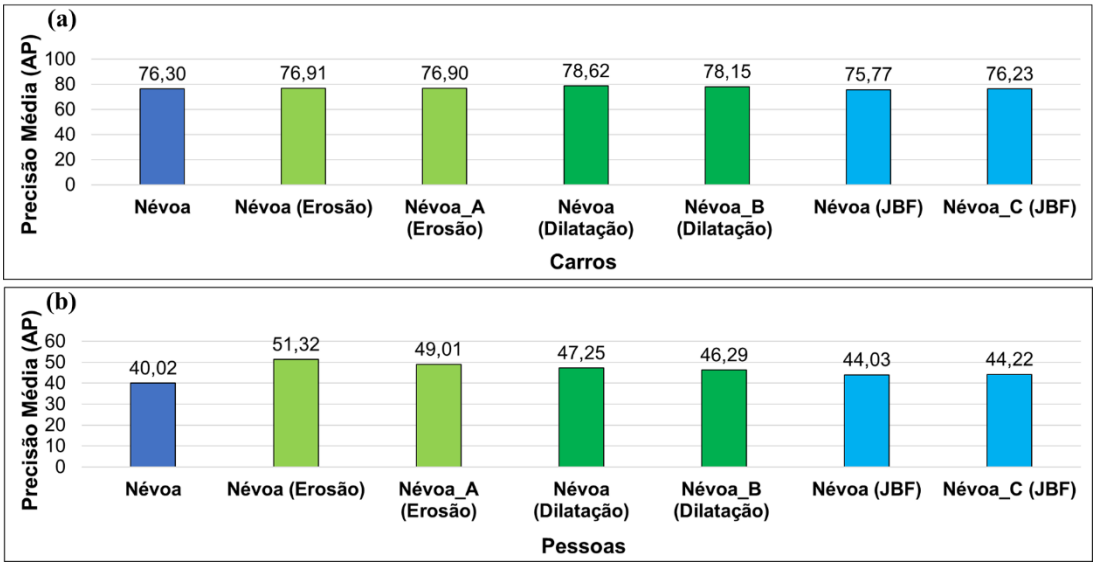

Figura 46 – Precisão média para o conjunto de imagens com névoa do DAWN para as classes de (a) Carros e (b) Pessoas

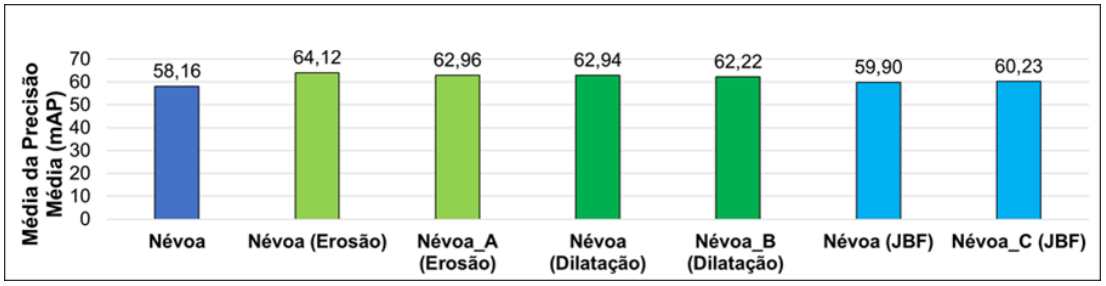

Figura 47 – mAP para o com o conjunto de imagens com névoa do DAWN

O depositório que contém imagens em condição de chuva foi identificado por ‘Chuva’ e o primeiro resultado representa o teste com o conjunto com a CNN treinada sem a aplicação de filtros. A CNN foi treinada com o filtro de Erosão e os testes foram realizados considerando o conjunto de ‘Chuva’ e as respectivas as imagens com o filtro de Erosão aplicado, identificadas como ‘Chuva_A’. A CNN também foi treinada com o filtro de Dilatação e testada com o conjunto de imagens ‘Chuva’ e a aplicação do filtro de Dilatação, denominado como ‘Chuva_B’. Finalmente, a CNN treinada com o filtro JBF foi testada com o conjunto ‘Chuva’ e com o conjunto ‘Chuva’, com o filtro JBF aplicado (Chuva_C).

Na Figura 48 (a) estão os resultados de AP para a classe de Carros e na Figura 48 (b) os resultados de AP para as classes de Pessoas. Os resultados dos testes em condição de Chuva foram semelhantes aos resultados em condição de Névoa. Os filtros que resultaram em melhor desempenho para a classe de Carros foram os de Erosão e Dilatação

e, similarmente, o filtro JBF diminuiu o desempenho. O mesmo ocorreu para a classe de Pessoas, mostrando que para este conjunto de imagens todos os filtros melhoraram o desempenho da CNN. Nota-se que aplicação dos filtros de Erosão, Dilatação e JBF geraram os melhores resultados, respectivamente.

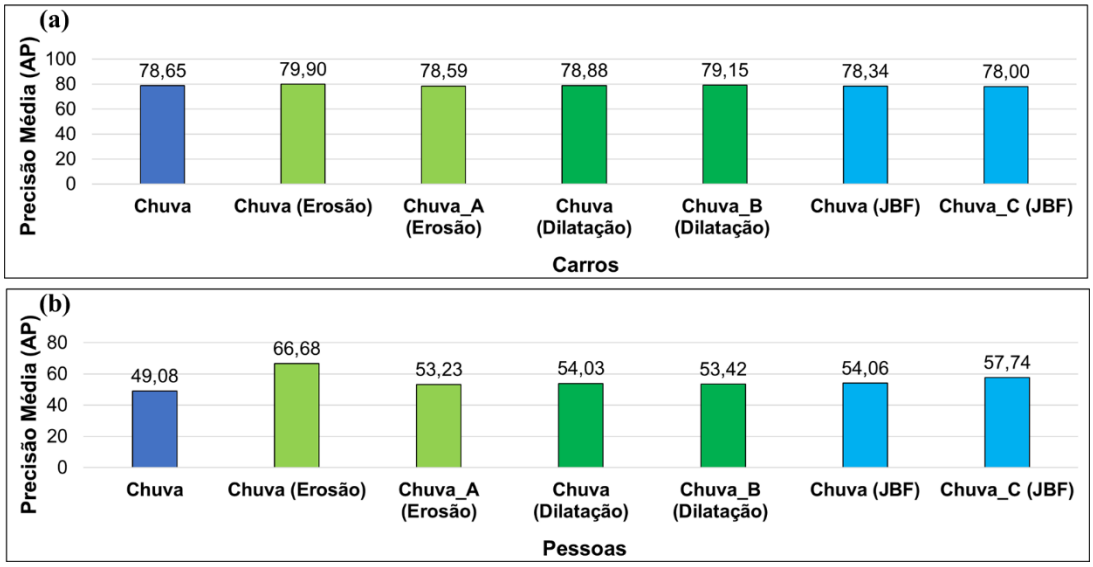

Figura 48 – Precisão média para o conjunto de imagens com chuva do DAWN para as classes de (a) Carros e (b) Pessoas

Na Figura 49 estão os resultados de mAP para o conjunto de imagens com chuva do DAWN. Para esta condição, identifica-se que a aplicação dos filtros melhorou o desempenho da CNN. O filtro de Erosão novamente apresentou os melhores resultados, seguido pelos filtros de Dilatação e JBF.

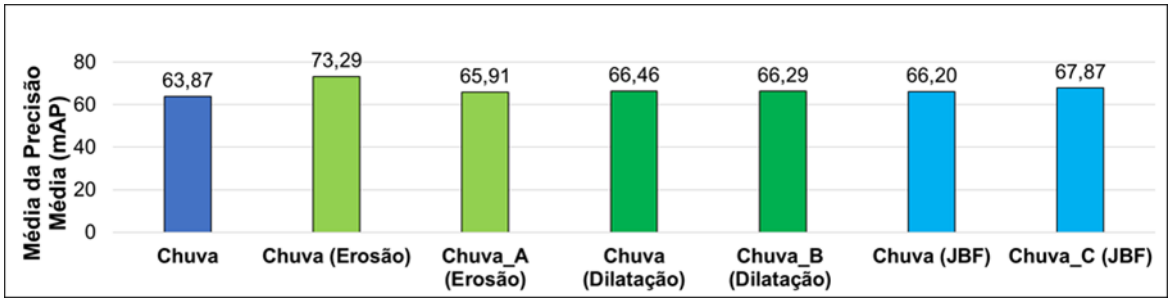

Figura 49 – mAP para o com o conjunto de imagens com chuva do DAWN

Na Figura 50 (a) se encontram os resultados de AP para a classe de Carros para as condições de neve, névoa e chuva, que é nomeado como o conjunto de imagens ‘DAWN’. Na Figura 50 (b), se encontram os resultados de AP para a classe de Pessoas considerado o conjunto ‘DAWN’. Os resultados de ‘DAWN’ se referem aos testes da CNN sem

utilização de filtros durante o treinamento. Na sequência, para a CNN treinada com o filtro de Erosão, têm-se os resultados dos testes com o conjunto ‘DAWN’ e as imagens do conjunto ‘DAWN’, com a aplicação do filtro de erosão, identificado como “DAWN_A”. Os resultados da CNN treinada com o filtro de Dilatação se encontram em seguida, no qual se identificam os testes com o conjunto ‘DAWN’ e o mesmo com a aplicação do filtro de Dilatação, nomeado como “DAWN_B”. Finalmente, a CNN treinada com o filtro JBF foi testada com o conjunto ‘DAWN’ e com o conjunto ‘DAWN’ com o filtro JBF aplicado (DAWN_C).

Nota-se que, para a classe de Carros, os filtros de Erosão e Dilatação demonstraram melhor desempenho em relação à condição sem a aplicação de filtros, elevando a métrica de AP em cerca de 1,2%. Por outro lado, o a aplicação do filtro JBF não aumentou a métrica de AP. Para a classe de Pessoas, somente a aplicação do filtro de Erosão representou aumento da métrica de AP, em torno de 3,0%. Contudo, os filtros de Dilatação e JBF não aumentaram o desempenho da CNN. Observa-se que para as duas classes os melhores resultados com a CNN treinada com os filtros foram em testes com as imagens do conjunto DAWN com a aplicação dos filtros.

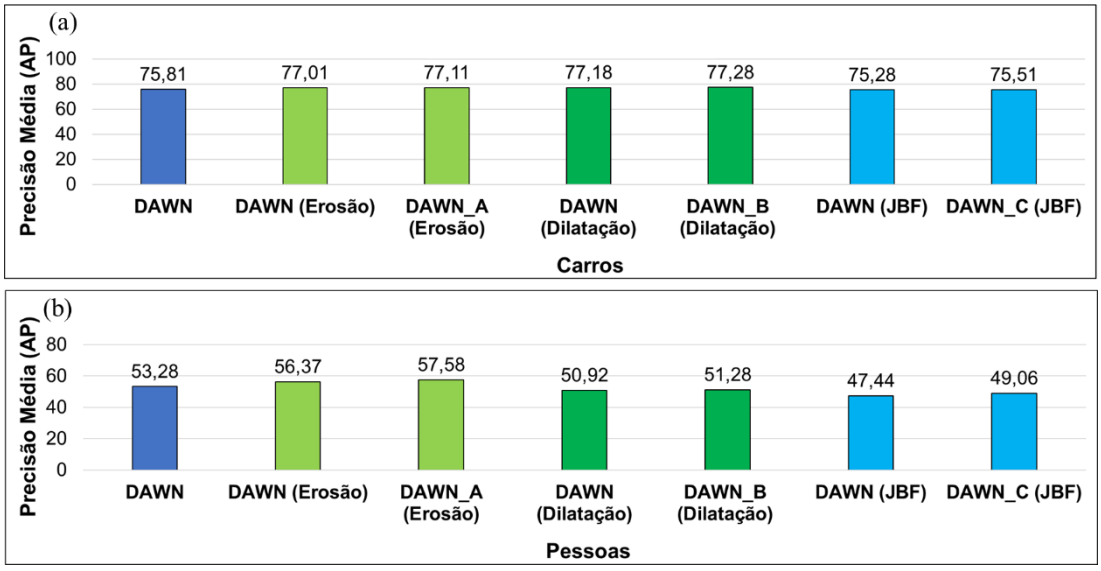

Figura 50 – Precisão média para o conjunto de imagens com chuva do DAWN para as classes de (a) Carros e (b) Pessoas

Na Figura 51, estão os resultados de mAP para o conjunto de imagens DAWN. Identifica-se uma melhora de cerca de 3% na métrica de mAP com a utilização do filtro de Erosão, em contrapartida que os demais filtros, Dilatação e JBF, não obtiveram melhora

na métrica de mAP. O melhor resulta se expressa com a utilização do filtro de Erosão durante o treinamento e nas imagens de testes (DAWN_A).

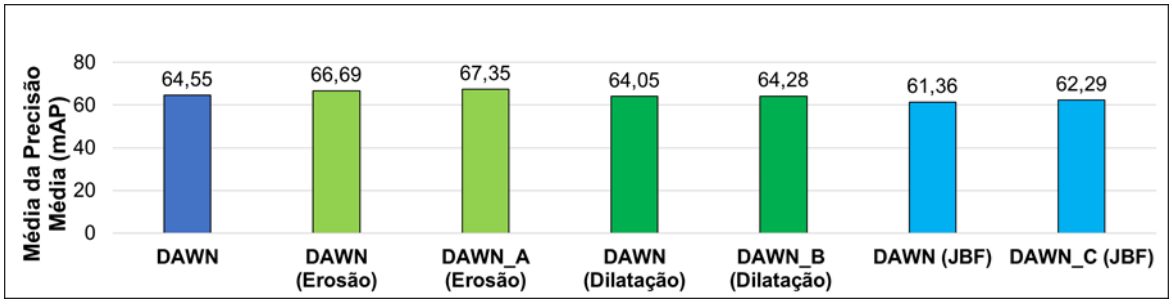

Figura 51 – mAP para os conjuntos de imagens de neve, névoa e chuva do DAWN

Nota-se que, nos resultados por condição climática e no conjunto DAWN, o filtro de Erosão demonstrou os melhores resultados, tanto com as imagens de testes originais e com a aplicação de filtros. Verifica-se que, para todas as condições em que se utilizou o filtro de Erosão durante o treinamento da CNN, a aplicação da filtragem nas imagens de testes aumentou o desempenho da CNN com relação as imagens de testes sem a utilização dos filtros verificando-se a métrica de mAP.

9 CONCLUSÃO E CONSIDERAÇÕES FINAIS

Neste trabalho, para a detecção de objetos em condições climáticas adversas, aplicou-se a biblioteca OpenCV durante o treinamento e testes da CNN YOLOv3 com o intuito de melhorar o seu desempenho em tais condições.

Primeiramente, treinou-se a CNN, com os recursos do Google Colab (versão gratuita) utilizando 70.000 imagens do conjunto BDD100K. Na sequência, o treinamento foi realizado com os recursos do Google Colab Pro. A despeito de o Google Colab Pro ter levado a tempos menores de treinamento, uma comparação direta de desempenho não é possível devido à seleção da GPU ser automática. Entretanto, notou-se que além da possibilidade da utilização de GPUs mais potentes o maior tempo de inatividade do Google Colba Pro contribuiu significativamente para melhor eficiência do treinamento.

Além disso, imagens de entrada de diferentes tamanhos durante o treinamento em 416 x 416, 512 x 512, 608 x 608 e 960 x 960 *pixels* demonstraram diferenças significativas nas métricas de avaliação. A influência do tamanho da imagem de entrada foi notável nos testes realizados com o conjunto BDD100K. Observou-se um aumento de cerca de 20% na métrica de mAP aumentando-se o tamanho de imagem de 416 x 416 *pixels* para 960 x 960 *pixels.* De forma complementar verificou-se que os melhores resultados estão condicionados à CNN testada com o respectivo tamanho de imagem considerado durante o treinamento. Pelo fato de o ambiente utilizado para avaliação ser virtual, a análise do tempo de processamento com o aumento do tamanho da imagem de entrada não foi considerada no escopo do trabalho. Isso poderá ser considerado em trabalhos futuros em conjunto com variáveis de dinâmica veicular.

Em relação aos testes com o conjunto DAWN, foi possível verificar o efeito das condições adversas na detecção de objetos. Em relação a análise dos resultados por classe de objeto a classe de pessoas apresentou resultado inferior à de veículos, pois é uma das classes desafiadoras para detecção pela CNN. Constatou-se adicionalmente que a condição climática que mais impactou foi a de neve, principalmente pelo fato de cobrir os objetos. Na sequência, os resultados que mais afetaram a detecção foram a névoa e a chuva, respectivamente.

A implementação de filtragem com OpenCV, para o treinamento e testes não contribuiu para o aumento do desempenho da CNN para o conjunto BDD100K. Em compensação, quando se compara os resultados de testes do conjunto DAWN com a

implementação dos filtros do OpenCV, é perceptível a melhora no desempenho. Avaliando-se separadamente por condição climática a métrica de mAP, o a aplicação do filtro de Erosão melhorou o desempenho da CNN. Os filtros de Dilatação e JBF somente não melhoraram os resultados na condição de Neve. Em relação ao conjunto DAWN, o melhor resultado aconteceu com a aplicação do filtro de Erosão durante o treinamento e os testes, com um aumento de cerca de 3% da mAP.

O trabalho desenvolvido mostra o potencial da utilização da biblioteca OpenCV durante treinamento e testes de CNN para detecção de objetos. Para trabalhos futuros, pode ser considerada a avaliação da detecção em condições críticas adversas com as limitações e requisitos impostos por um sistema automotivo real.

10 REFERÊNCIAS

AGGARWAL, C. C. The Basic Architecture of Neural Networks. [s.l: s.n.].

AZIZ, L. et al. Exploring deep learning-based architecture, strategies, applications and current trends in generic object detection: A comprehensive review. IEEE Access, v. 8, p. 170461–170495, 2020.

BASU, S., A. BANERJEE, R. MOONEY. Semi-supervised clustering by seeding. In Proceedings of the Nineteenth International Conference on Machine Learning - ICML-2002, Sidney, Australia, pp. 19-26.

BENGIO, Y. Learning Deep Architectures for AI. Foundations, v. 2, p. 1–55, 2009.

BABA, A., PASHA, M. G., AHAMMED, S. A., TABASSUM, S. N. Introduction to Neural Networks Design Architecture. International Journal of Scientific & Engineering Research, 2013.

BALLARD, D. H.; HINTON, G. E.; SEJNOWSKI, T. J. Parallel visual computation. Nature, v. 306, n. 5938, p. 21–26, 1983. ISSN 1476-4687. Disponível em: <https: //doi.org/10.1038/306021a0>.

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

BRAGA, A. d. P.; CARVALHO, A. P. d. L. F.; LUDERMIR, T. B. Redes Neurais Artificiais: Teoria e Aplicações. Rio de Janeiro: LTC Editora, 2000. 262 p. ISBN 9788521615644. Disponível em: <https://books.google.com.br/books?id=R-p1GwAACAAJ>.

BROWNLEE, J. A Gentle Introduction to the Rectified Linear Unit (ReLU). 2019a. Disponível em: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/. Acesso em: 23 de fevereiro de 2023.

BROWNLEE, J. How to Configure the Learning Rate When Training Deep Learning Neural Networks. 2019b. Disponível em: https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/. Acesso em: 26 de fevereiro de 2023.

BRUCE, R. A bayesian approach to semi-supervised learning. In M. Ishisuka & A. Satter (Eds), Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium - NLPRS-2001, Tokyo, Japão, pp. 57-64.

Centers For Disease Control And Prevention (CDC). Road Traffic Injuries and Deaths - A Global Problem. 2020. Disponível em: https://www.cdc.gov/injury/features/global-road-safety/index.html. Acesso em: 10 de dezembro de 2022.

CARVALHO, A. Redes Neurais Artificiais. 2009. Disponível em: https://sites.icmc.usp.br/andre/research/neural/. Acesso em: 09 de janeiro de 2023.

CHEN, Y. et al. Domain Adaptive Faster R-CNN for Object Detection in the Wild. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, p. 3339–3348, 2018.

CYGERT, S.; CZYZEWSKI, A. Toward robust pedestrian detection with data augmentation. IEEE Access, v. 8, p. 136674–136683, 2020.

DAI, J. et al. R-FCN: Object detection via region-based fully convolutional networks. Advances in Neural Information Processing Systems, p. 379–387, 2016.

DALAL, N. TRIGGS, B. Histograms of oriented gradients for human detection. vol. 1. IEEE. 2005.

DATA SCIENCE ACADEMY. As 10 principais arquiteturas de redes neurais. In: . Deep Learning Book. Brasília, DF: Data Science Academy, 2019. cap. 10. Disponível em: <http://deeplearningbook.com.br/as-10-principais-arquiteturas-de-redes-neurais/>. Acesso em: 17 de janeiro de 2023

DING, Q. et al. CF-YOLO: Cross Fusion YOLO for Object Detection in Adverse Weather with a High-quality Real Snow Dataset. v. 14, n. 8, p. 1–10, 2022.

DECKER, K. M., FOCARDI, S. 1995. Technology overview: A report on data mining. Technical Report CSCS TR-95-02, CSCS-ETH, Swiss Scientific Computing Center.

ERAZO, J. Desenvolvimento de um Sistema de Contagem e Classificação de Veículos Utilizando Redes Neurais Convolucionais. Dissertação (Mestrado) - Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina. 2021.

EVERINGHAM, M. et al. The pascal visual object classes (VOC) challenge. International Journal of Computer Vision, v. 88, n. 2, p. 303–338, 2010.

FRANCO, C. R. Inteligência Artificial. Londrina: Uniasselvi, 2014. 168 p. ISBN 978-85-68075-77-7.

FELZENSZWALB, P; D. MCALLESTER; D. RAMANAN. A discriminatively trained, multiscale, deformable part model. IEEE. 2008.

GEIGER, A. et al. Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research. The International Journal of Robotics Research, n. October, p. 1–6, 2013.

GOODFELLOW, I., BENGIO, Y., COURVILLE, A. Deep Learning. MIT Press. 2016. Disponível em: <http://www.deeplearningbook.org/>. Acesso em: 17 de janeiro de 2023.

GONZALEZ R. C., WOODS, R. E. Digital image processing, 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 2002.

HARIS, M.; GLOWACZ, A. Road object detection: a comparative study of deep learning-based algorithms. Multimedia Tools and Applications, 2022.

HE, K. et al. Deep residual learning for image recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, v. 2016-Decem, p. 770–778, 2016.

HNEWA, M.; RADHA, H. Object Detection under Rainy Conditions for Autonomous Vehicles: A Review of State-of-the-Art and Emerging Techniques. IEEE Signal Processing Magazine, v. 38, n. 1, p. 53–67, 2021.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 32nd International Conference on Machine Learning, ICML 2015, v. 1, p. 448–456, 2015.

KAMASSURY, J. Decodificação de Códigos de Comprimento Curto usando Redes Neurais Profundas. Dissertação de Mestrado em Engenharia Elétrica na Universidade Federal de Santa Catarina, 2020.

KOK, J. N. et al. Artificial Intelligence: Definition, Trends, Techniques and Cases. Encyclopedia of Life Support Systems (EOLSS), p. 1096–1097, 2010.

KENK, M. A.; HASSABALLAH, M. DAWN: Vehicle Detection in Adverse Weather Nature Dataset. p. 1–6, 2020.

KOPF, J. et al. Joint bilateral upsampling. ACM Transactions on Graphics, v. 26, n. 3, 2007.

KRIZHEVSKY; A., SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. 2012.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. Nature, v. 521, n. 7553, p. 436–444, 2015. ISSN 1476-4687. Disponível em: <https://doi.org/10.1038/nature14539>.

LI, Yuqi. Colab Pro vs. Free — AI Computing Performance. 2021. Disponível em: https://towardsdatascience.com/colab-pro-vs-free-ai-computing-performance-4e983d578fb2. Acesso em: 26 de outubro de 2022.

LIN, T. Y. et al. Microsoft COCO: Common objects in context. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 8693 LNCS, n. PART 5, p. 740–755, 2014.

LIN, T. Y. et al. Feature pyramid networks for object detection. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, v. 2017- Janua, p. 936–944, 2017.

LIN, T. Y. et al. Focal Loss for Dense Object Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 42, n. 2, p. 318–327, 2020.

LIU, Wei et al. SSD: Single shot multibox detector. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), [s. l.], v. 9905 LNCS, p. 21–37, 2016.

LIU, M. Y.; BREUEL, T.; KAUTZ, J. Unsupervised image-to-image translation networks. Advances in Neural Information Processing Systems, v. 2017-Decem, n. Nips, p. 701–709, 2017.

LIU, L., OUYANG, W., WANG, X., FIEGUTH, P., CHEN, J., LIU, X., PIETIKÄINEN, M. Deep Learning for Generic Object Detection: A Survey. 2018.

LIU, L. et al. Deep Learning for Generic Object Detection: A Survey. International Journal of Computer Vision, v. 128, n. 2, p. 261–318, 2020. ISSN 1573-1405. Disponível em: <https://doi.org/10.1007/s11263-019-01247-4>.

LIU, B.; SU, S.; WEI, J. The Effect of Data Augmentation Methods on Pedestrian Object Detection. Electronics (Switzerland), v. 11, n. 19, p. 1–17, 2022.

MARENGONI, M.; STRINGHINI, S. Tutorial: Introdução à Visão Computacional usando OpenCV. Revista de Informática Teórica e Aplicada, v. 16, n. 1, p. 125–160, 2010.

MATHWORKS. 2022. Disponível em: https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html. Acesso em: 10 de dezembro de 2022.

MITCHELL, T. M. et al. 1997. Machine learning. WCB.

M. EVERINGHAM, L. V. GOOL, C. WILLIAMS, J. WINN, AND A. Z. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. v. 2012, p. 1–32, 2012.

MACQUEEN, J. B. 1967. Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. pp. 281–297.

MCCALLUM, A., K. NIGAM, & L. H. UNGAR. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In R. Ramakrishnan, S. Stolfo, R. Bayardo, & I. Parsa (Eds.), Proceedinmgs of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining — KDD-00, New York, USA, pp. 169–178. ACM Press.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. Foundations of Machine Learning. 2. ed. New York, USA: MIT Press, 2018. ISBN 0262039400.

NEUHOLD, G. et al. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. Proceedings of the IEEE International Conference on Computer Vision, v. 2017-Octob, p. 5000–5009, 2017.

NEXAR. 2022. Disponível em: https://data.getnexar.com/. Acesso em: 15 de novembro de 2022.

NILSSON, N. J. Introduction to Machine Learning: An Early Draft of a Proposed Textbook. 1998. 179 p. Disponível em: <https://ai.stanford.edu/~nilsson/MLBOOK.pdf>. Acesso em: 07 de janeiro de 2023.

OPENCV. 2022. Eroding and Dilating. Disponível em: https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html. Acesso em: 10 de dezembro de 2022.

OPEN SOURCE. 2023. The 2-Clause BSD License. Disponível em: https://opensource.org/license/bsd-2-clause/. Acesso em: 23 de abril de 2023.

PATIL, A.; RANE, M. Convolutional Neural Networks: An Overview and Its Applications in Pattern Recognition. Smart Innovation, Systems and Technologies, v. 195, p. 21–30, 2021.

PETSCHNIGG, G. et al. Digital photography with flash and no-flash image pairs. ACM SIGGRAPH 2004 Papers, SIGGRAPH 2004, p. 664–672, 2004.

PYTHON. 2023. Generate pseudo-random numbers. Disponível em: https://docs.python.org/3/library/random.html. Acesso em: 26 de fevereiro de 2023.

REDMON, J. et al. You only look once: Unified, real-time object detection. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, v. 2016- Decem, p. 779–788, 2016.

REDMON, J.; FARHADI, A. YOLO9000: Better, faster, stronger. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, v. 2017-Janua, p. 6517–6525, 2017.

REDMON, J.; FARHADI, A. YOLO v.3. Tech report, p. 1–6, 2018.

REN, S. et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 39, n. 6, p. 1137–1149, 2017.

RUSSELL et al, 2008. LabelMe: a database and web-based tool for image annotation. International Journal of Computer Vision, v. 77, p. 157-173.

RUSSELL, S.; NORVIG, P. Artificial Intelligence: A Modern Approach. 3rd. ed. USA: Prentice Hall Press, 2009. ISBN 0136042597

SASAKI, Y. The truth of the F-measure. Teach Tutor mater, p. 1–5, 2007.

SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V., RABINOVICH, A. Going Deeper with Convolutions. In the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

SHARMA, V.; RAI, S.; DEV, A. A Comprehensive Study of Artificial Neural Networks. International Journal of Advanced research in computer science and software engineering, 2012.

SCHMIDHUBER, J. Deep Learning in Neural Networks: An Overview. Neural networks, 61:85–117. 2015.

SILVA, W. Estimação da Densidade de Multidões: Proposta de Arquitetura. Dissertação (Mestrado) - Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Goiás. 2021.

SUMMERS, C.; DINNEEN, M. J. Improved mixed-example data augmentation. Proceedings - 2019 IEEE Winter Conference on Applications of Computer Vision, WACV 2019, p. 1262–1270, 2019.

TOMASI, C.; MANDUCHI, R. Bilateral Filtering for Gray and Color Images. Proceedings of the International Conference on Computer Vision, Bombay, 1998.

TURAY, T.; VLADIMIROVA, T. Toward Performing Image Classification and Object Detection with Convolutional Neural Networks in Autonomous Driving Systems: A Survey. IEEE Access, v. 10, p. 14076–14119, 2022.

VALIATI, G. R. Uma especialização do YOLOv3 para Detecção de Pedestres. Dissertação (Mestrado) - Programa de Pós-Graduação em Informática, Universidade Federal do Paraná. 2019.

VILLÁN, A. F. Mastering OpenCV 4 with Python.

VIOLA, P; JONES, M. Rapid object detection using a boosted cascade of simple features. vol. 1. IEEE. 2001.

YU, D.; JI, S. A New Spatial-Oriented Object Detection Framework for Remote Sensing Images. IEEE Transactions on Geoscience and Remote Sensing, v. 60, 2022.

YU, F. et al. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, p. 2633–2642, 2020.

ZHANG, S.; BENENSON, R.; SCHIELE, B. CityPersons: A diverse dataset for pedestrian detection. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, v. 2017- Janua, p. 4457–4465, 2017.

ZHAO, Z.; ZHENG, P.; TAO, S.; WU, X. Object Detection with Deep Learning: A Review. IEEE transactions on neural networks and learning systems. 2019.

ZOU, Z. et al. Object Detection in 20 Years: A Survey. Proceedings of the IEEE, p. 1–22, 2023.