RODRIGO CARVALHO BARBOSA

CLASSIFICAÇÃO DE VEÍCULOS BASEADA EM DEEP LEARNING PARA APLICAÇÃO EM SEMÁFOROS INTELIGENTES

LAVRAS – MG

2021

RODRIGO CARVALHO BARBOSA

CLASSIFICAÇÃO DE VEÍCULOS BASEADA EM DEEP LEARNING PARA APLICAÇÃO EM SEMÁFOROS INTELIGENTES

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Redes de Computadores e Sistemas Embarcados, para a obtenção do título de Mestre.

Prof. DSc. Demóstenes Zegarra Rodríguez

Orientador

Profa. DSc. Renata Lopes Rosa

Coorientadora

LAVRAS – MG

2021

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Barbosa, Rodrigo Carvalho.
Classificação de veículos baseada em Deep Learning para aplicação em semáforos inteligentes / Rodrigo Carvalho Barbosa. - 2021.
87 p. : il.

Orientador(a): Prof. DSc. Demóstenes Zegarra Rodríguez.
Coorientador(a): Profa. DSc. Renata Lopes Rosa.
Dissertação (mestrado acadêmico) - Universidade Federal de Lavras, 2021.
Bibliografia.

1. Classificação por imagens. 2. Deep Learning. 3. You Only Look Once. I. Rodríguez, Demóstenes Zegarra. II. Rosa, Renata Lopes III. Título.

RODRIGO CARVALHO BARBOSA

CLASSIFICAÇÃO DE VEÍCULOS BASEADA EM DEEP LEARNING PARA APLICAÇÃO EM SEMÁFOROS INTELIGENTES

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Redes de Computadores e Sistemas Embarcados, para a obtenção do título de Mestre.

APROVADA em 19 de Novembro de 2020.

Prof. MSc. Bruno de Abreu Silva UFLA
Prof. DSc. João Carlos Giacomin UFLA

Prof. DSc. Demóstenes Zegarra Rodríguez
Orientador

Profa. DSc. Renata Lopes Rosa
Co-Orientadora

LAVRAS – MG
2021

Dedico a realização deste sonho aos meus pais que me ajudaram e apoiaram para realizar este feito em minha vida, aos meus colegas de curso que me ajudaram em momentos de dificuldade e principalmente ao Prof. DSc. Demóstenes Zegarra Rodríguez e a Profa. DSc. Renata Lopes Rosa pela paciência de ambos comigo, compreensão, ajuda e apoio, pois no meio desta pesquisa até o fim passei por vários problemas pessoais.

AGRADECIMENTOS

Agradeço primeiramente ao Dr. Demóstenes Zegarra Rodríguez, pela dedicada orientação, disponibilidade, compreensão e apoio, sempre com paciência comigo, à Dra. Renata Lopes Rosa pela ajuda para a realização dos treinamentos e testes desta dissertação.

A minha família maravilhosa que, mesmo diante das dificuldades, sempre me apoiaram com paciência, compreensão e amor, mesmo com dificuldades financeiras, foram fundamentais a esta conquista. A minha namorada Thaynara Michelle Martins Inácio, em breve será minha noiva e esposa, que sempre esteve do meu lado e, principalmente, nos momentos difíceis, me ajudando e apoiando.

Ao amigo e colega de mestrado, Cairo Aparecido Campos, pela amizade, apoio, dedicação, paciência e companheirismo nas aulas, no Restaurante Universitário (RU), risadas e dificuldades, durante todo o curso, além do apoio que me deu forças para poder continuar com o mestrado.

A todos os amigos do mestrado, pela amizade e aprendizado compartilhados, especialmente, ao amigo Diego Vinícius Natividade pelo conhecimento compartilhado, nas aulas de sistemas operacionais e nesta reta final da minha dissertação, companheirismo e sua generosidade.

E principalmente à Pós Graduação da UFLA do curso de Ciência da Computação e todo seu corpo docente, onde eu tive total apoio e obteve toda a estrutura necessária para a realização dos testes e treinamentos para que fosse possível eu poder concluir este mestrado, o qual eu sonho há mais de oito anos. O meu muito obrigado a todos!

RESUMO

Atualmente, na literatura, muitos estudos surgem a cada momento a fim de diminuir a intervenção humana e melhorar a qualidade de vida, propondo novos serviços, mecanismos por meio de aplicativos, inovações tecnológicas e sensores automáticos. Visando à mobilidade urbana, os semáforos de trânsito são serviços que são explorados. Algoritmos de *Deep Learning (DL)* vêm sendo muito utilizados para identificação e classificação de imagens para tomada de decisão no trânsito, com o objetivo de detectar veículos de segurança e saúde pública. Porém, faltam algoritmos para classificar imagens em semáforos inteligentes com alta precisão e de resposta rápida. Nesta pesquisa, é proposto um sistema de detecção de imagens para diferentes tipos de serviços como veículos de segurança, saúde e transporte público e veículos comuns, integrado a um semáforo inteligente. Além disso, também é proposto um algoritmo de priorização baseado no Código de Trânsito Brasileiro (CTB). O sistema de detecção é embasado em um algoritmo *DL*, usando um modelo aprimorado do *You Only Look Once (Version 3) (YOLOv3)*, que foi chamada de Rede de identificação de veículos prioritários (PVInet, do inglês *Priority Vehicles Identification Network*). Além disso, é proposta uma estratégia de design para o modelo *PVInet* que apresenta um alto desempenho em termos de tempo de execução. Para o treinamento do modelo *PVInet*, foi criado um novo Banco de Dados (BD) que considera imagens homogêneas de veículos do trânsito Brasileiro, uma vez que os atuais BD disponível na literatura são de imagens heterogêneas. A proposta da nossa solução pode ajudar a reduzir o tempo de espera de veículos com prioridade nas estradas controladas por um semáforo, algo que não ocorre quando comparado a um semáforo atual com tempos de espera fixos.

Palavras-chave: Classificação por imagens. *Deep Learning*. *You Only Look Once*. Semáforo Inteligente. Código de Trânsito Brasileiro.

ABSTRACT

Currently, in the literature, many studies appear every moment in order to reduce human intervention and improve the quality of life, proposing new services, mechanisms through applications, technological innovations and automatic sensors. Aiming at urban mobility, traffic lights are services that are exploited. Algorithms of DL have been widely used for identification and classification of images for decision making in traffic, with the objective of detecting safety and public health vehicles. However, there is a lack of algorithms to classify images into intelligent traffic lights with high precision and quick response. In this research, an image detection system is proposed for different types of services such as security, health and public transport vehicles and common vehicles, integrated with an intelligent traffic light. In addition, a prioritization algorithm based on CTB is also proposed. The detection system is based on a DL algorithm, using an improved model from YOLOv3, which was called the Priority Vehicle Identification Network (PVInet). In addition, a design strategy for the PVInet model is proposed, which presents a high performance in terms of execution time. For the training of the PVInet model, a new BD was created that considers homogeneous images of Brazilian traffic vehicles, since the current BD available in the literature are heterogeneous images. Our solution proposal can help to reduce the waiting time for vehicles with priority on roads controlled by a traffic light, something that does not happen when compared to a current traffic light with fixed waiting times.

Keywords: Classification by images. Deep Learning. You Only Look Once. Smart Semaphore. Brazilian Traffic Code.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

BD Banco de Dados

BN Bayesian Network

CB Connection Block

CTB Código de Trânsito Brasileiro

CNN Convolutional Neural Network

Di Direita

DL Deep Learning

DBN Deep Belief Networks

DenseNet Densely Connected Convolutional Networks

ELU Exponencial Linear Units

Es Esquerda

Fast R-CNN Fast Region-based Convolutional Neural Network

Faster R-CNN Faster Region-based Convolutional Neural Network

FDDB Face Detection Data Set and Benchmark

FN False Negative

FPS Frames Per Second

FP False Positive

Fr Frontal

GPU Graphics Processing Unit

IA Inteligência Artificial

IoT Internet of Things

ITS Intelligent Transport System

LReLU Leaky Rectified Linear Unit

LSTM Long Short-Term Memory

mAP mean Average Precision

PReLU Parametric Rectified Linear Unit

PIR Passive Infrared Sensor

PVInet Priority Vehicles Identification Network

RN Redes Neurais

RNA Rede Neural Artificial

R-CNN Regions with Convolutional Neural Network

R-FCN Region-based Fully Convolutional Networks

RPN Region Proposal Network

ReLU Rectified Linear Unit

RReLU Randomized Leaky Rectified Linear Unit

SRS Soft-Root-Sign

SD-SRCNN Simple Denoising and Super-Resolution Convolutional Neural Network

SSD Single Shot Detector

SJ Subcategoria Junta

SGD Stochastic Gradient Descent

SELU Scaled Exponential Linear Unit

TIC Tecnologias de Informação e Comunicação

TN True Negative

TP True Positive

VANETs Vehicular Ad-hoc Networks

VOC07 Visual Object Classes 2007

VOC12 Visual Object Classes 2012

YOLO You Only Look Once

YOLOv2 You Only Look Once (Version 2)

YOLOv3 You Only Look Once (Version 3)

YOLOv2-tiny You Only Look Once (tiny Version 2)

YOLOv3-tiny You Only Look Once (tiny Version 3)

SUMÁRIO

1 INTRODUÇÃO

Congestionamento no tráfego urbano é um problema mundial, principalmente por parte da população, pois alteram suas agendas pessoais, podem causar problemas na economia como o atraso de entrega de mercadorias e até certas doenças respiratórias causadas pelas emissões de gases poluentes emitidos pelos carros (BAUZA; GOZALVEZ; SANCHEZ-SORIANO, 2010).

Com o avanço de pesquisas e o surgimento de novas tecnologias, visando facilitar a vida e o conforto dos humanos, houve uma grande evolução nos veículos nas últimas décadas. As cidades onde habitamos estão seguindo este mesmo ritmo de mudanças. Exemplos destas mudanças é o surgimento do conceito de cidades inteligentes, no qual toda a infraestrutura da cidade onde habitamos é inteligente e interligada entre si. Cidades como Amsterdã, Barcelona, *Copenhagen, Helsinki, Manchester* e *Vienna* são exemplos deste conceito (MANVILLE et al., 2014).

Estas cidades usam vários mecanismos de infraestrutura inteligente. Um exemplo deste tipo de mecanismo é um modelo de estacionamento inteligente, proposto por ALAM et al. (2018), no qual esta solução utiliza o uso de câmeras em um Sistema Inteligente de Transporte *Intelligent Transport System (ITS)* distribuído, que consiste em um sistema de estacionamento com visão computacional em que a Inteligência Artificial (IA) é distribuída e as decisões são descentralizadas.

Na literatura atual, os veículos autônomos estão em bastante evidência. Várias pesquisas estão sendo desenvolvidas e implementadas. Atualmente, várias multinacionais famosas na área de tecnologia, estão criando protótipos de carros autônomos. Estes carros têm o propósito de serem dirigidos sem a intervenção humana.

Um exemplo deste tipo de protótipo é a proposta da BOSH BEHRENDT, NOVAK e BOTROS (2017). Esta solução consiste em um sistema de visão computacional com *Deep Learning (DL)* para detecção, rastreamento e localização em 3D de semáforos. Muitos destes veículos usam *Machine Learning* ou aprendizado de máquina, onde dos quais utilizam computadores com *Graphics Processing Unit (GPU)*.

Na literatura, existem vários tipos de algoritmos para detecção de objetos como o *Regions with Convolutional Neural Network (R-CNN)* GIRSHICK et al. (2014), o *Fast Region-based Convolutional Neural Network (Fast R-CNN)* GIRSHICK (2015), *You Only Look Once (YOLO)* REDMON et al. (2016), *You Only Look Once (Version 2) (YOLOv2)* REDMON e FARHADI (2017), *Single Shot Detector (SSD)* LIU et al. (2016) ou versões mais modernas do *YOLO* como

o caso do *YOLOv3* REDMON e FARHADI (2019), os quais muitas vezes são utilizados para detecção de placas, pessoas, objetos na estradas ou vias e a aproximação com outros veículos.

Veículos autônomos utilizam estes tipos de algoritmos, porém a infraestrutura no trânsito em grandes centros urbanos também vem sendo modificada, utilizando outras abordagens com o uso da IA, visando o melhoramento do tráfego urbano. Assim, o investimento em semáforos inteligentes para tentar diminuir congestionamento e acidentes de trânsito é muito relevante. Esses semáforos utilizam IA, para reconhecimento de imagens do trânsito.

Em CHOUDEKAR, BANERJEE e MUJU (2011), os autores propuseram processamento de imagens em tempo real. Nos testes, eles verificaram que para, imagens escala de cinza, o tempo de resposta era mais rápido que em imagens coloridas. Em MOGHADDAM, HOSSEINI e SAFABAKHSH (2015), os autores sugeriram o uso de lógica *fuzzy* para reduzir o tempo de espera em um cruzamento, assim eles conseguiram reduzir em até 10% o tempo de espera. Em LIU, LIU e CHEN (2017), os autores propuseram um projeto de semáforo que utiliza *Internet of Things (IoT)*, no qual utilizaram um simulador, usando o mapa da *Sunnyvale*, Califórnia. Em NELLORE e HANCKE (2016), os autores sugeriram um semáforo no qual dê prioridade para veículos de resgate ou de segurança pública através de sensores sonoros junto com o uso de imagens, onde capta as imagens no trânsito e identifica o veículo, sua velocidade e a distância até chegar ao semáforo. Em ZAID, SUHWEIL e YAMAN (2017), os autores propuseram um projeto de semáforo, onde ele adquire as imagens do trânsito, prepara as imagens, segmenta, utiliza uma Rede Neural Artificial (RNA), a qual classifica os veículos, retorna o número de veículos na via e por meio de uma lógica *fuzzy*, irá determinar o tempo de espera para a mudança de um estado do semáforo para outro.

Em WU e TSAI (2016), os autores propuseram um projeto para classificação de pedestres, bicicletas, motocicletas e veículos por meio do DL usando a técnica de *Deep Belief Networks (DBN)* (HINTON, 2009). Após a realização dos testes, a acurácia chegou a 89,53%. Em LEE e CHUNG (2017), os autores tiveram uma taxa de precisão de 92,98% classificando essas mesmas classes. Eles utilizaram uma junção de técnicas de DL, como *AlexNet* RUSSAKOVSKY et al. (2014), *GoogLeNet* SZEGEDY et al. (2015) e *ResNet18* (HE et al., 2016).

É importante destacar que os algoritmos de DL são mais utilizados em projetos de veículos autônomos. Em BEHRENDT, NOVAK e BOTROS (2017) e PON et al. (2018), os autores propuseram um sistema de detecção de semáforos e placas de trânsito ao mesmo tempo, preen-

chendo uma lacuna na literatura, pois não há um conjunto de dados de treinamento que combine estas duas situações.

Muitas soluções para detecção de imagens em tempo real, que utilizam o DL estão utilizando o modelo de arquitetura *SSD* LIU et al. (2016) ou o *YOLO* (REDMON et al., 2016). Uma destas soluções é a do MÜLLER e DIETMAYER (2018), em que se propõe detecção de semáforos utilizando o *SSD*, na solução o tempo de resposta obteve resultados satisfatórios assim como sua acurácia. Segundo GU et al. (2017) o uso de *SSD* tem resultados satisfatórios devido à velocidade de detecção e a sua precisão. O uso do *YOLOv3* ZHANG e ZHU (2019) tem um tempo de resposta mais rápido e com alta acurácia para detectar objetos além de estar bastante consolidado na literatura.

Para a extração de recursos, o *YOLOv3* utiliza o *Darknet-53* o qual consiste em 53 camadas, sendo que as versões anteriores utilizavam o *Darknet-18* com 18 camadas. A principal característica deste modelo de arquitetura consiste em detecção de vários objetos em uma imagem, todo o processo de extração de característica da imagem é realizado somente uma vez.

Apesar dos avanços no desenvolvimento de semáforos inteligentes, os quais utilizam sensores como de presença ou sonoros, alguns destes sensores podem gerar falsos dados para a entrada no algoritmo, causando um efeito de erros em cascata. Além disso, o custo para a instalação e manutenção pode ser um pouco elevados, o que muitas vezes inviabiliza a implementação destes semáforos em um cenário real. Além disso, a acurácia alcançada pelos atuais algoritmos de detecção e classificação de diversos tipos de veículos, especialmente os de emergência, ainda devem ser melhorados para obter um sistema de transito confiável. Nesse mesmo sentido, é importante que o tempo de processamento dos algoritmos deve também ser reduzido.

De acordo com o CTB Cidades (2008), quem tem a maior prioridade no trânsito são os veículos de segurança pública, resgate e/ou de saúde, desde que estejam com a luz sinalizadora de emergência mais conhecida como giroflex acionado e com sua devida adesivação no veículo. Caso contrário, os pedestres e os ciclistas deverão ter prioridade respectivamente no trânsito devido a sua fragilidade e vulnerabilidade no trânsito.

Nesse contexto, espera-se que sistemas de semáforo inteligente possam identificar e classificar veículos de emergência como polícia, ambulância e bombeiros, para diminuir o tempo de espera no trânsito. Para que esse tipo de semáforos tenha sucesso, é importante obter valores altos de acurácia na identificação e classificação desses veículos.

1.1 Justificativa

A principal justificativa do presente trabalho é melhorar o desempenho dos algoritmos de identificação e classificação de diferentes veículos de trânsito, por meio de imagens, para usar em semáforos inteligentes. Utilizamos uma proposta de melhoramento do *YOLOv3* e, para isso, foi construído um algoritmo denominado como Rede de Identificação de Veículos Prioritários (*PVInet*, do inglês *Priority Vehicles Identification Network*), que consiste em um grande melhoramento no quesito de classificação em comparação com o *YOLOv3*.

1.2 Objetivos

O principal objetivo deste trabalho é propor um modelo de detecção e classificação de imagens de veículos chamado *PVInet*, o qual foi baseado no algoritmo *YOLOv3*, que consiste em reduzir o número de parâmetros do modelo do *YOLOv3*, mantendo um desempenho de detecção desejável, através da implementação de uma estrutura de conexão densa aprimorada.

O *PVInet* demonstra um melhor desempenho em acurácia e tempo de processamento em comparação com outros trabalhos similares. Para diminuir este tempo utilizamos as três funções de ativação, que foram a *Softmax*, *Rectified Linear Unit (ReLU)* e a *Soft-Root-Sign (SRS)*, sendo que a última obteve melhores resultados.

Propomos também a construção de um modelo de algoritmo de semáforo inteligente, no qual a saída do *PVInet* será a entrada de um modelo de semáforo inteligente.

1.2.1 Objetivos específicos

- Construir uma base de dados (BD) com dados homogêneos, ou seja, imagens com as mesmas características como resolução, dimensão e tamanho para cada classe de veículo. Além disso, três ângulos de toma de imagem são considerados. Deve-se destacar que na literatura não se conta com uma BD com características similares;
- Testar diferentes funções de ativação no algoritmo proposto *PVInet* com o intuito de diminuir o tempo de execução do algoritmo;
- Propor um algoritmo de semáforo inteligente, que utilize como entrada a saída do *PVInet*, sendo que os veículos com maior prioridade são os de emergência;

- Utilizar a função de ativação mais recente na literatura a SRS e assim comprovar a sua eficácia em comparação a outras funções de ativação;
- Realizar vários testes e treinamentos usando o *YOLOv3* original e diversas versões de algoritmos derivados do *YOLOv3*, neste caso o *PVInet*.

1.3 Estrutura da Dissertação

No capítulo a seguir, será abordado os conceitos fundamentais para o entendimento deste trabalho. No Capítulo 3, os trabalhos relacionados, tanto em desenvolvimento, quanto já desenvolvidos, serão descritos. No Capítulo 4, a metodologia adotada. No Capítulo 5, os resultados e por fim, no Capítulo 6, será abordada a conclusão da dissertação.

2 REFERENCIAL TEÓRICO

No referencial serão apresentados uma breve descrição da história e origem dos semáforos, conceitos de cidades inteligentes, uma abordagem de diferentes algoritmos de inteligência artificial e DL.

2.1 Um breve histórico dos semáforos

Segundo HOLANDA (1986), a palavra semáforo se origina da junção de duas palavras gregas que é a *sema* (sinal) e *phoros* (que leva), semáforos (sinal que leva).

Antigamente, no meio dos cruzamentos existia uma guarita onde os policias eram responsáveis em controlar o tráfego de veículos nas vias. Com o passar dos anos, começaram a surgir semáforos com circuitos eletrônicos, que são os semáforos atuais, porém os primeiros semáforos funcionavam com gás natural.

O primeiro semáforo com circuitos eletrônicos parecido aos atuais, constituía num modelo acionado por um policial em um poste no cruzamento, o protótipo foi apresentado em 1912 e foi implementado em 1914. Somente em 1920 um policial chamado *William Potts* criou o semáforo que conhecemos hoje em dia com 3 cores que são: VERMELHO para parar, o AMARELO para atenção e o VERDE para seguir.

2.2 Cidades inteligentes

Nos últimos tempos, o conceito de "cidade inteligente" vem crescendo cada vez mais, na literatura científica ou em pesquisas e até mesmo em cidades que estão adotando estes modelos. Em um futuro breve, as cidades irão ter uma atuação essencial sejam em aspetos sociais, econômicos ou ambientais para todos os que habitam estas cidades. (MORI; CHRISTODOULOU, 2012).

De acordo com dados da UNICEF, a população mundial no ano de 2017 era de quase 7,6 bilhões de pessoas. A maior parte desta população mundial se concentravam na Ásia com 60% (4,5 bilhões), seguido da África com 17% (1,3 bilhões), Europa com 10% (742 milhões), América Latina e Caribe com 9% (646 milhões), na América do Norte (361 milhões) e na Oceania (41 milhões). China (1,4 bilhão) e Índia (1,3 bilhão) continuam sendo os dois países mais populosos do mundo, compreendendo 19% e 18% do total global, respectivamente (WHO; MATHERS et al., 2017). A Tabela 2.1, mostra uma perspectiva estimada da população mundial.

Table 2.1 – População Mundial nos Próximos Anos

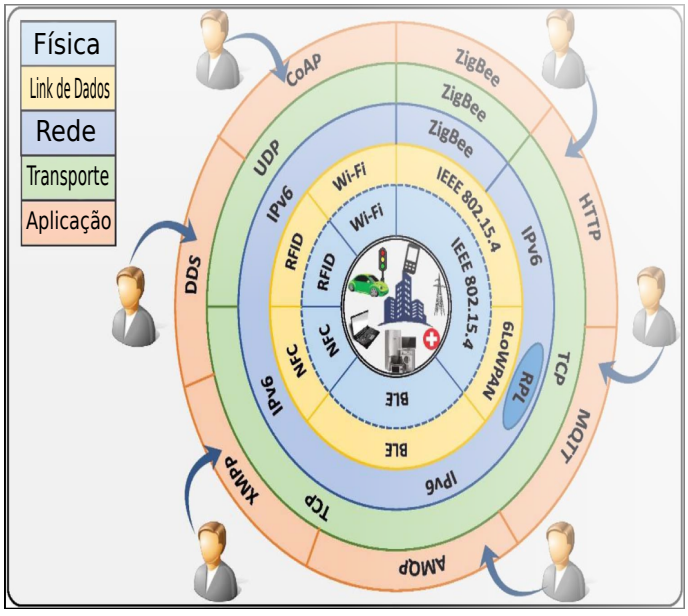
Região	População (Milhões)			
	ANO			
	2017	**2030**	**2050**	**2100**
Mundo	7 550	8 551	9 772	11 184
África	1 256	1 704	2 528	4 468
Europa	742	739	716	65
América Latina e Caribe	646	718	780	712
América do Norte	361	395	435	499
Oceania	41	48	57	72

Fonte: Adaptado de (WHO; MATHERS et al., 2017)

Estima-se que em até 2030, o número de veículos no mundo poderá superar dois bilhões (SPERLING; GORDON, 2008). De acordo com as estimativas até lá, já teremos veículos elétricos com valores mais acessíveis para a população como um todo.

Para uma cidade ser considerada uma cidade inteligente, os habitantes devem ter total interação com tudo o que a cidade tem a oferecer, como o uso de aplicativos de celular (CEBALLOS; LARIOS, 2016). Por exemplo, pagar uma passagem de ônibus através do celular sem a necessidade de um cartão ou dinheiro em mãos. Tecnologia e interatividade entre cidade e habitante é um ponto crucial em uma cidade inteligente, como é demonstrado na Figura 2.1.

Figure 2.1 – Topologia de uma cidade inteligente



Fonte: Adaptado de (SILVAA, 2018)

Segundo CHOURABI et al. (2012), para que uma cidade inteligente funcione sem nenhum problema, a infraestrutura da cidade depende inteiramente de canais de fibra óptica, redes

Wi-Fi, pontos de acesso sem fio de qualidade e sistemas de informações orientados a serviços. Já é possível termos cidades inteligentes graças a uma série de tecnologias que já usamos e algumas que estão tendo uma alta gama de pesquisa.

Podemos citar o uso de rede 4G, como tecnologia atual. As redes ad-hoc veiculares Vehicular Ad-hoc Networks (VANETs) como tecnologia já sendo utilizadas em algumas cidades e carros autônomos como uma tecnologia do futuro. Segundo KUMAR, GOEL e MALLICK (2018), uma cidade inteligente abrange várias áreas, que vai desde a área governamental até a área de entretenimento de uma cidade.

2.2.1 Cidades modelo

Algumas cidades, como é o caso das cidades de Malta, Nova York, Amsterdam, Rio de Janeiro e Barcelona; estão investindo um valor elevado em tecnologias novas, preparando suas cidades para um dia se tornarem uma cidade inteligente. Estas cidades em questão visam melhorar os serviços de finanças, educação, turismo, atendimento hospitalar e vários setores de entretenimento (ANGELIDOU, 2014).

Os indicadores mais comuns para uma cidade inteligente ou cidade modelo são (GIFFINGER RUDOLF; FERTNER, 2007) :

- Economia Inteligente (Competitividade): Espírito inovador, empreendedorismo, imagem e marcas econômicas, produtividade, flexibilidade do mercado de trabalho, incorporação internacional e capacidade de transformar;
- Pessoas Inteligentes (Capital Social e Humano): Nível de qualificação, afinidade com a aprendizagem ao longo da vida, pluralidade social e étnica, flexibilidade, criatividade e participação na vida pública;
- Governança Inteligente (Participação): Participação na tomada de decisões, serviços públicos e sociais, governança transparente, estratégias e perspectivas políticas;
- Mobilidade Inteligente (Transporte e Tecnologias de Informação e Comunicação Tecnologias de Informação e Comunicação (TIC)): Acessibilidade local; Acessibilidade (internacional) nacional; disponibilidade de infraestrutura de TIC, sistemas de transporte sustentáveis, inovadores e seguros;

- Ambiente Inteligente (Recursos naturais): Atratividade em condições naturais, proteção ambiental, gerenciamento sustentável de recursos;
- Vida Inteligente (Qualidade de vida): Instalações culturais, condições de saúde, segurança individual, qualidade de moradia, instalações educacionais, atratividade turística e Coesão social;

2.3 Inteligência artificial

No decorrer dos séculos, com o surgimento da ciência, vários cientistas realizaram pesquisas para compreender a inteligência humana. Após várias pesquisas, surgiu um campo de estudos que podemos chamar de IA.

A IA é o ramo da ciência da computação que propõe elaborar determinados dispositivos que simulam a capacidade do ser humano ser inteligente (RUSSELL; NORVIG, 2020). Estudos para IA são recentes se compararmos a outros campos.

As primeiras pesquisas para este campo da ciência começaram no ano de 1956, logo após a Segunda Guerra Mundial, por isso as pesquisas sobre IA têm muito campo a ser explorado, muitas pesquisas a serem realizadas, assim aplicar a IA em várias outras áreas de pesquisas.

Apesar da literatura sobre o assunto ser relativamente nova, existe uma gigantesca variedade de subcampos para pesquisa, que vai desde uma máquina para jogar xadrez com um humano, até uma máquina para conseguir identificar o rosto humano e liberar o acesso de uma determinada área através da biometria da iris da pessoa (RUSSELL; NORVIG, 2020).

2.3.1 Panorama da inteligência artificial

No ano de 1943 McCulloch & Pitts publicaram no Bulletion of Mathematical Biophysics, Vol. 5, p. 115-133, como eles desenvolveram um modelo de neurônio artificial onde era possível uma máquina aprender (MCCULLOCH; PITTS, 1943). No ano de 1950 *Alan Turing* escreveu um artigo sobre máquinas capazes de jogar Xadrez. O nome do artigo que o deixaria famoso foi Mind (SMITH CHRIS; MCGUIRE; YANG, 2006). No verão do ano de 1956 aconteceu o *Dartmouth Summer Research Project on Artificial Intelligence* onde *John McCarthy, Marvin Minsky, Nathaniel Rochester* e *Claude Shannon* propuseram o termo IA (MCCARTHY JOHN; MINSKY; SHANNON, 2006).

Porém, na década de 1970 até 1980, os pesquisadores de IA viveram a década do que hoje podemos chamar de a década ártica para a IA, pois as pesquisas esbarravam em dois pontos cruciais, falta de hardware e falta de dados para o uso de uma IA (RUSSELL; NORVIG, 2020).

Em 1981, os japoneses anunciaram um projeto de hardware, com o objetivo de escrever programas e construir máquinas que pudessem conversar, traduzir idiomas, interpretar imagens e raciocinar como seres humanos (MCCORDUCK, 2004). Para este projeto, eles utilizaram o prolog como linguagem primária (CREVIER, 1993).

Com medo de domínio japonês, países da Europa e EUA, investiram em várias pesquisas relacionadas a IA (TOSCANO, 2009).

Devido a estes investimentos das grandes potências mundiais em pesquisas, hoje há uma variedade de campos e subcampos relacionados a IA, como demonstração e resolução de teoremas matemáticos, criação de poesias e músicas, diagnóstico de doenças e até mesmo na direção de um veículo em uma estrada movimentada. A IA, também pode ser utilizada em qualquer tarefa intelectual, podendo substituir o homem em várias atividades do nosso dia a dia (RUSSELL; NORVIG, 2020).

2.3.2 Aprendizado de máquina

Aprendizado de máquina ou mais conhecido do inglês - *machine learning* - é uma área da ciência da computação que surgiu graças aos subcampos de reconhecimento de padrões e IA.

Nos últimos anos, o Aprendizado de Máquina tornou-se um dos pilares da tecnologia da informação e com isso, uma parte central, embora usualmente oculta de nossa vida. Com a crescente quantidade de dados disponíveis, há boas razões para acreditar que a análise inteligente de dados se tornará ainda mais intensa como um ingrediente necessário para o progresso tecnológico (SMOLA; VISHWANATHAN, 2008).

2.3.3 Tipos de Aprendizado de máquina

Em termos genéricos, podemos classificar o aprendizado de máquina em dois principais subcampos que é o aprendizado supervisionado e o aprendizado não supervisionado.

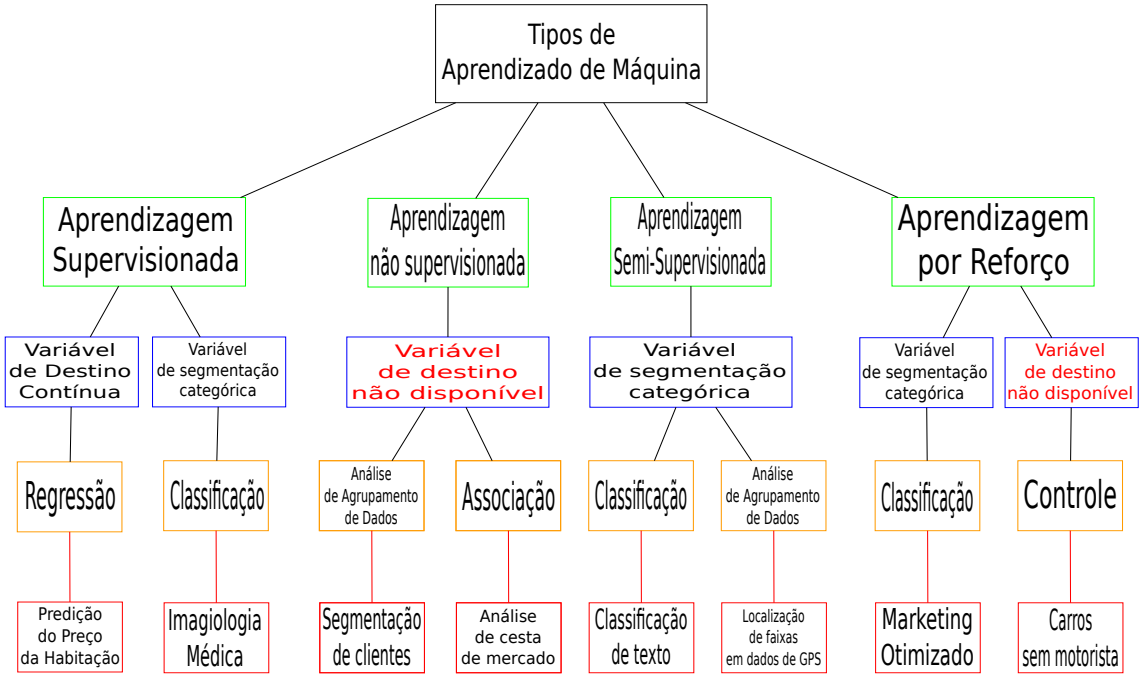
No aprendizado supervisionado o objetivo é aprender uma regra geral que mapeia as entradas para as saídas. No entanto, no aprendizado não supervisionado nenhum tipo de entrada

é dado ao algoritmo de aprendizado, deixando-o sozinho para encontrar estrutura nas entradas fornecidas (BARBER, 2012).

Além do aprendizado supervisionado e o não supervisionado, existem mais dois subcampos que são o aprendizado semi-supervisionado e o aprendizado por reforço onde cada um se divide em mais subcampos.

A Figura 2.2, ilustra os tipos de treinamento para Aprendizado de Máquina, que são aprendizagem supervisionada, aprendizagem não supervisionada, aprendizagem semi-supervisionada e aprendizagem por reforço (os tipos de aprendizagem serão explicados posteriormente) e os últimos quadros mostram alguns para cada tipo de aplicação.

Figure 2.2 – Tipos de Aprendizado de Máquina



Fonte: Autor (2020)

2.3.3.1 Aprendizado supervisionado

O objetivo do aprendizado supervisionado é aprender um mapeamento de entrada para uma saída cujos valores corretos são fornecidos por um gestor. Já, no aprendizado não supervisionado não existe este tal gestor e tem apenas os dados de entrada. Destaca-se que seu objetivo é encontrar as regularidades de entrada (ALPAYDIN, 2010).

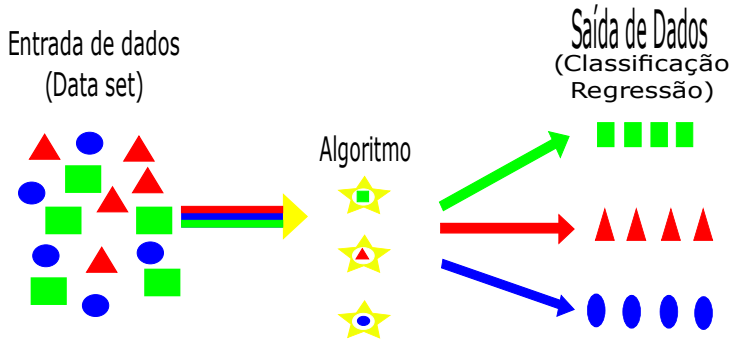
A aprendizagem supervisionada é bastante comum em problemas de classificação e regressão (ZHANG, 2010) (MOHRI; ROSTAMIZADEH; TALWALKAR, 2018). Em um pro-

blema de regressão, estamos tentando prever os resultados em uma saída contínua. Em um problema de classificação, estamos tentando prever os resultados em uma saída discreta.

O aprendizado supervisionado é quando o modelo aprende por meio de uma supervisão a partir de resultado pré-definidos ou rotulados.

A Figura 2.3, ilustra um exemplo de aprendizado supervisionado.

Figure 2.3 – Aprendizado Supervisionado

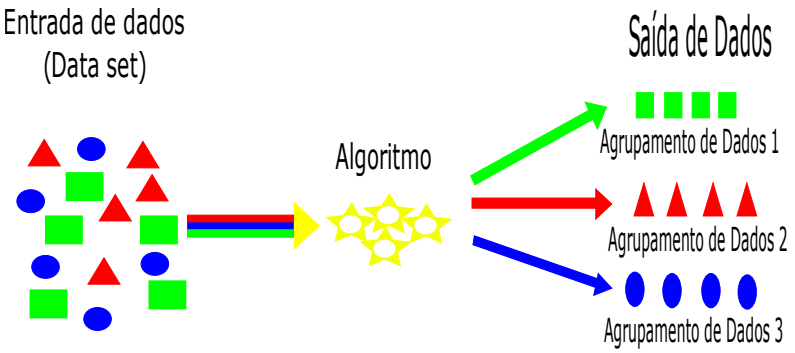


Fonte: Autor (2020)

2.3.3.2 Aprendizado não supervisionado

Exemplos de utilização do modelo de aprendizagem não supervisionado mais comuns são o agrupamento e associação (MOHRI; ROSTAMIZADEH; TALWALKAR, 2018) (RUSSELL; NORVIG, 2020). A Figura 2.4, ilustra um exemplo de aprendizado não supervisionado. O aprendizado não supervisionado é quando o modelo aprende sem uma supervisão, dando resultados por meio de agrupamentos ou associação.

Figure 2.4 – Aprendizado não Supervisionado



Fonte: Autor (2020)

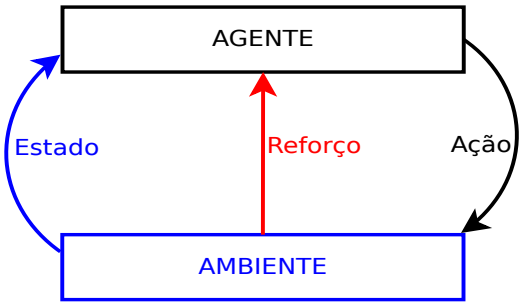
2.3.3.3 Aprendizado semi-supervisionado

O aprendizado semi-supervisionado é o intermediário entre o aprendizado supervisionado e o aprendizado não supervisionado. Além das informações não avaliadas, o algoritmo é fornecido com algumas referências supervisionadas (CHAPELLE; SCHÖLKOPF; ZIEN, 2010). Classificação e agrupamentos são exemplos de aplicações deste modelo (RUSSELL; NORVIG, 2020).

2.3.3.4 Aprendizado por reforço

Aprendizado por reforço, é aquele em que um agente está inserido em um ambiente e interage com ele através de percepções e ações, conforme a Figura 2.5. O agente recebe uma entrada e uma indicação do estado, com isso ele escolhe, uma ação a tomar e gera uma saída. A ação altera então o estado do ambiente e à medida que essa mudança de estado é informada, ao agente através de um valor de sinal de reforço (ALPAYDIN, 2010). No aprendizado por reforço, o agente aprende com uma série de reforços, recompensas ou punição. Classificação e controle são exemplos de aplicações deste modelo (RUSSELL; NORVIG, 2020).

Figure 2.5 – Modelo padrão de aprendizado por reforço



Fonte: Autor (2020)

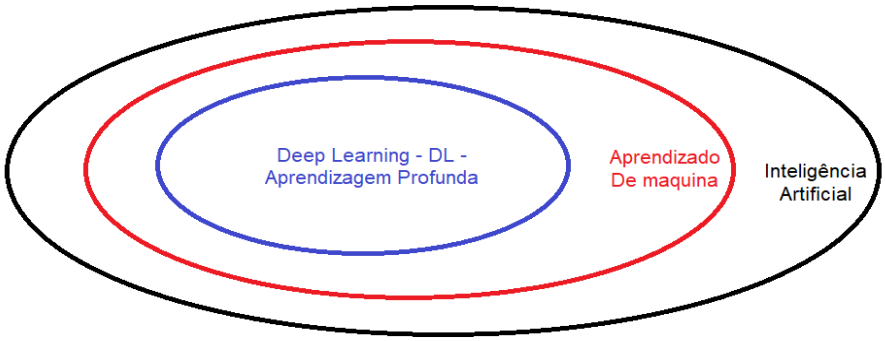
2.3.4 *Deep Learning*

Desde 2006, o aprendizado estruturado profundo, ou mais conhecido como DL ou aprendizado hierárquico, mostrou-se como uma nova área de pesquisa em aprendizado de máquina. No decorrer dos anos, vários métodos novos foram apresentados a partir de análises empregando o DL, onde está gerando um número bastante considerável de trabalhos no processamento de sinais e informações, alterando a forma com que estes dados eram processados criando novas técnicas e o apoio das mídias no qual estão divulgando sua ascensão desde o surgimento.

No geral, o modelo atual do DL envolve um número relativamente alto de camadas sucessivas de representação, nas quais todas as camadas são aprendidas automaticamente quando os dados de treinamento são inseridos. Isso é o grande o diferencial do DL, pois outras abordagens de aprendizado de máquina tendem a se concentrar em aprender uma camada ou duas camadas de representações dos dados (CHOLLET, 2018).

O DL pode ser usado para tarefas de aprendizado de máquina supervisionadas e não supervisionadas (BENGIO, 2009). A Figura 2.6, ilustra a hierarquia do DL desde a inteligência artificial.

Figure 2.6 – Conceitos fundamentais



Fonte: Autor (2020)

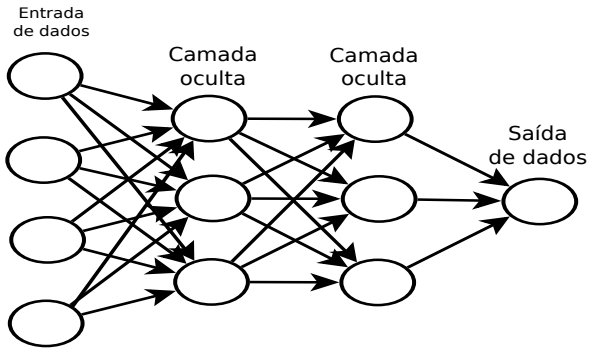
Na literatura atual, o DL vem sendo muito utilizado em propostas em diversas áreas como, processamento de linguagem natural, recuperação de informações, análise de vídeos, visão computacional, detecção de anomalias, sistema de recomendação, reconhecimento de padrões, detecção de objetos, entre outras coisas (SCHMIDHUBER, 2015).

2.3.4.1 Principais arquiteturas

Usualmente, redes neurais artificiais são organizadas em "camadas". As quais, dependendo do número, podem ser de "camada única"ou "camada múltipla". A arquitetura de um DL, vai depender dos tipos de conexões entre os neurônios. Existe dois tipos principais, que são “redes neurais *Feed-Forward*” e “redes neurais recorrentes” (SAZLI, 2006).

- **Rede Neural *Feed-Forward*:** Se a rede não tiver um retorno das saídas dos neurônios para as entradas de outros neurônios, então podemos considerar que é uma (rede neural *Feed-Forward*). A Figura 2.7, ilustra como funciona uma rede feed-forward.

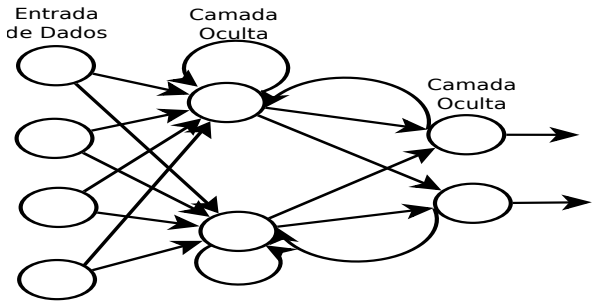
Figure 2.7 – Arquitetura *Feed-Forward*



Fonte: Adaptado de (CIABURRO, 2017)

– **Rede Neural Recorrente:** Se a rede tiver um retorno das saídas dos neurônios para as entradas de outros neurônios, ou seja, a saída do próprio neurônio e a entrada dele, então podemos considerar que esta rede é uma rede neural recorrente. A Figura 2.8, ilustra como funciona uma rede Rede Neural Recorrente.

Figure 2.8 – Funcionamento de Rede Neural Recorrente



Fonte: Adaptado de (MULDER STEVEN BETHARDB, 2015)

Na atual literatura, existem várias discussões sobre um termo para DL. Assim ocasionalmente, podemos ouvir o termo "redes neurais da nova geração". Um bom exemplo dos modelos com uma arquitetura profunda são Redes neurais de feed-forward ou Multilayer Perceptron (MLP) com muitas camadas ocultas (DENG, 2014). Nos últimos anos, o número de arquiteturas e algoritmos utilizados no DL está bastante amplo e variado. A Tabela 2.2, mostra algumas arquiteturas, o ano que eles apareceram e aplicações com o DL. Apesar de *Long Short-Term Memory (LSTM)* e *Convolutional Neural Network (CNN)* serem relativamente antigas, ambas são arquiteturas bastante usadas hoje em dia.

Table 2.2 – Principais arquiteturas

Época que surgiu	Arquitetura	Aplicações
1990 a 1995	RNN *(Recurrent Neural Network)*	Reconhecimento de fala, manuscrito
1995 a 2000	*LSTM (Long short-term memory)*	Compressão de texto em linguagem natural, reconhecimento de manuscrito, reconhecimento de fala, reconhecimento de gestos, legenda de imagens
1995 a 2000	CTB *(Convolutional Neural Network)*	Reconhecimento de imagem, análise de vídeo, processamento de linguagem
2005 a 2010	DBN *(Deep Belief Network)*	Reconhecimento de imagem, recuperação de informação, compreensão da linguagem natural, previsão de falhas
2010 a 2015	DSN *(Deep Sparse-coded Network)*	Recuperação de informação, reconhecimento de fala contínuo
2015	GRU *(Gated Recurrent Unit)*	Mesmas aplicações que a arquitetura *LSTM*

Fonte: Autor (2020)

2.4 Funções de ativação

De acordo com LECUN, BENGIO e HINTON (2015), uma função de ativação controla a propagação da informação através de camadas adjacentes para redes neurais profundas e fornece a propriedade não linear. Assim sendo, a função de ativação é crucial para o desempenho de RNA e o comportamento de aprendizagem (ZHOU et al., 2020).

Na literatura existem várias funções de ativação, a função ReLU é a mais conhecida e utilizada em várias propostas usando o DL, porém existem algumas desvantagens. Porém a função SRS, supera estas desvantagens e vem obtendo resultados satisfatórios.

2.4.1 Função Softmax

A função *Softmax*, é também conhecida como Softargmax GOODFELLOW, BENGIO e COURVILLE (2016) ou função exponencial normalizada (BISHOP, 2006). De acordo com NWANKPA et al. (2018), a função *Softmax* é usada para calcular a distribuição de probabilidade

de um vetor de números reais. Essa função produz uma saída que é uma faixa entre 0 e 1, com a soma das probabilidades sendo igual a 1. O cálculo para a função *Softmax* GOODFELLOW, BENGIO e COURVILLE (2016), é representado pela função 2.1.

$$f_{(x_i)} = \frac{exp\,(x_i)}{\sum_j exp\,(x_j)} \tag{2.1}$$

Esta função aparece em várias camadas de saída em diferentes modelos de arquiteturas de *DL*, onde são usadas (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) (BADRINARAYANAN; KENDALL; CIPOLLA, 2017).

A função calcula o exponencial de cada resultado das camadas ocultas da rede do DL. Após esse cálculo, são somados todos estes resultados. Em seguida, cada resultado do exponencial é divido pelo o resultado desta soma.

2.4.2 Função ReLU

A função ReLU foi proposta por (NAIR; HINTON, 2010). Segundo (NWANKPA et al., 2018). Essa função de ativação não linear ReLU tem sido amplamente utilizada para várias aplicações com *DL*. Esta função faz um calculo de limite para cada elemento de entrada onde os valores menores que 0 são definidos como 0 (NWANKPA et al., 2018). O cálculo da função *ReLU*, é dado pela função2.2.

$$f\,(x) = max\,(0,x) = \begin{cases} x_i \;\; if\, x_i \;\geq\; 0 \\ 0 \;\; if\, x_i \;<\; 0 \end{cases} \tag{2.2}$$

A função ReLU vem sendo usada dentro das unidades ocultas do *DL*. Ela é utilizada nas camadas de saída da rede com exemplos típicos encontrados em classificação de objetos KRIZHEVSKY, SUTSKEVER e HINTON (2012), HE et al. (2015) e aplicações de reconhecimento de voz (MAAS; HANNUN; NG, 2013).

Esta função da como resultado de 0 para valores negativos e mantém os valores acima de 0.

2.4.3 Função SRS

Mesmo a função de ativação ReLU sendo muito utilizada em camadas ocultas de algoritmos de detecção do DL, o algoritmo apresenta vários pontos negativos como: **Média diferente de zero, ausência de valores negativos e produção ilimitada** (ZHOU et al., 2020).

Na literatura atual, várias propostas de funções de ativação surgiram para suprir as deficiências do *ReLU* como a *Leaky Rectified Linear Unit (LReLU)* MAAS, HANNUN e NG (2013), *Parametric Rectified Linear Unit (PReLU)* HE et al. (2015), *Randomized Leaky Rectified Linear Unit (RReLU)* XU et al. (2015), *Exponencial Linear Units (ELU)* CLEVERT, UNTERTHINER e HOCHREITER (2015), *Scaled Exponential Linear Unit (SELU)* KLAMBAUER et al. (2017), RAMACHANDRAN, ZOPH e LE (2017) e a função *Maxout* GOODFELLOW et al. (2013), porém nenhuma delas conseguiram suprir os todas as deficiências da função.

Pretendendo superar estas desvantagens ZHOU et al. (2020) elaborando uma função de ativação não linear chamada de SRS, que é uma função suave, limitada e não monotônica. O que torna a função SRS se adaptar a um par de parâmetros treináveis para que o resultado seja melhor no desempenho de generalização, deve-se fornecer uma saída média zero e a velocidade de aprendizagem mais rápida. Da mesma forma que impede a saída seja espalhada no espaço de número real não negativo e a corrige para número real positivo, o que torna a função menos sensível ao iniciar e mais compatível com a normalização em lote.

Segundo ZHOU et al. (2020) uma função de ativação efetiva tem que ter: 1) Para acelerar o aprendizado é necessário ter valores negativos e positivos para controlar a média em direção a zero CLEVERT, UNTERTHINER e HOCHREITER (2015); 2) Para garantir um estado robusto a ruído, regiões de saturação (derivadas próximas de zero) e 3) Generalização eficaz e uma curva diferencial contínua que ajuda na otimização (RAMACHANDRAN; ZOPH; LE, 2017).

A função de ativação do *SRS* é definida por 2.3.

$$SRS(t) = \frac{t}{\frac{t}{\alpha} + e^{-\frac{1}{\beta}}} \tag{2.3}$$

onde as variáveis α e β representam um par de parâmetros positivos treináveis. O *SRS* apresenta uma região não monotônica em que $t < 0$ fornece a propriedade de média zero. Quando $t < 0$, ele evita e retifica a distribuição de saída. A derivada *SRS* é definida por 2.4.

$$SRS'(t) = \frac{\left(1 + \frac{t}{\beta}\right) e^{-\frac{t}{\beta}}}{\left(\frac{t}{\alpha} + e^{-\frac{t}{\beta}}\right)^2} \tag{2.4}$$

O *SRS* é uma saída limitada, apresentando o intervalo $\frac{\alpha\beta}{\beta - \alpha e}, \alpha$.

3 TRABALHOS RELACIONADOS

Nesta seção são apresentados os principais algoritmos de DL, de dois e de um estágio, as soluções baseadas no algoritmo *YOLO*, soluções de semáforo utilizando IA, e finalmente o CTB.

3.1 Algoritmos do DL

Nos últimos anos, foram apresentados vários modelos de algoritmos para DL, cada um com sua particularidade e meios de utilização. Dentre estes modelos podemos citar as redes *CNN* que é a base de vários modelos de algoritmos atuais.

3.1.1 Dois estágios

Os modelos de algoritmos, que usam o *CNN* como base, são métodos de detecção de objetos em dois estágios, foram os pioneiros e contém contribuições significativas na detecção de objetos. Estes algoritmos apresentam precisão na detecção de objetos, porém não são utilizados em aplicações em tempo real, pois demoram muito tempo para a detecção e classificação.

3.1.2 CNN

O nome *CNN* aponta que a rede usa uma operação matemática chamada convolução. Esta convolução é um tipo especializado de operação linear. As redes convolucionais são simplesmente Redes Neurais (RN) que usam convolução no lugar da multiplicação geral da matriz em pelo menos uma de suas camadas (GOODFELLOW; BENGIO; COURVILLE, 2016).

O *CNN* é um algoritmo de *DL* com várias camadas convolucionais, camadas de pool e camadas totalmente conectadas, o qual resultou em muitos avanços no reconhecimento e classificação de objetos em imagens, reconhecimento de voz, processamento de linguagem natural, são alguns exemplos de onde a *CNN* pode ser aplicada (RAZAVIAN et al., 2014) (VALUEVA et al., 2020).

- **Camada de convolução:** A camada de convolução é a primeira camada a extrair recurso de uma imagem de entrada. Ela preserva a relação entre os pixels, aprendendo os recursos da imagem usando pequenos quadrados de dados de entrada. Todo este processo realiza operações matemáticas a qual leva duas entradas, como matriz de imagem e um filtro ou kernel.

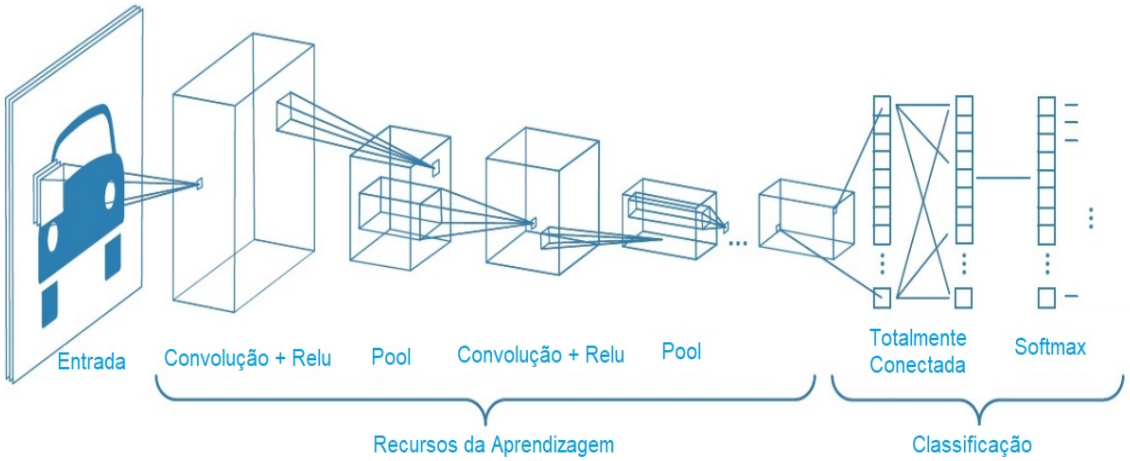
  A ReLU permite um treinamento mais rápido e eficaz, mapeando os valores negativos para zero e mantendo os valores positivos. Isso às vezes é chamado de ativação , porque apenas os recursos ativados são transportados para a próxima camada.

- **Camada Pool:** Após os dados saírem da camada de convolução, a camada pool faz um tipo especial de filtro à sua saída, onde as duas formas mais comuns são o pooling máximo e médio. Com o pooling máximo, é como aplicar um filtro máximo à imagem, já o pool médio aplica um filtro médio à imagem.

- **Camadas totalmente conectadas:** Apos a camada pool é produzido um vetor dimensional N , no qual N é o número de classes que o programa deve escolher. Este vetor contém as probabilidades de cada classe de qualquer imagem que está sendo classificada.

  A camada final da arquitetura *CNN* usa uma camada de classificação com a função de ativação Softmax para fornecer a saída de classificação. A Figura 3.1, ilustra como é o funcionamento do algoritmo do modelo *CNN*.

Figure 3.1 – Algoritmo *CNN*



Fonte: Autor (2020)

As equações do algoritmo *CNN* pode ser definida com um arranjo espacial de células acopladas localmente e um sistema dinâmico com entrada e saída (CHEN; HE; CHEN, 2006). As equações de estado do algoritmo *CNN* padrão (CHUA, 1998) são descritas pela equação 3.1.

$$\frac{d_{x_{i,j}}}{dt} = -x_{i,j} + z + \sum_{C_{k,l \in S_{i,j}}} + a_{k,l} y_{i+k,j+l} + \sum_{C_{k,l \in S_{i,j}}} b_{k,l} u_{i+k,j+l} \tag{3.1}$$

onde, $i, j \in Z^2$, com a equação de saída definida pela equação 3.2.

$$y_{i,j} = f\left(x_{i,j}\right) = \frac{1}{2}\left(\left|x_{i,j}+1\right| - \left|x_{i,j}-1\right|\right) \tag{3.2}$$

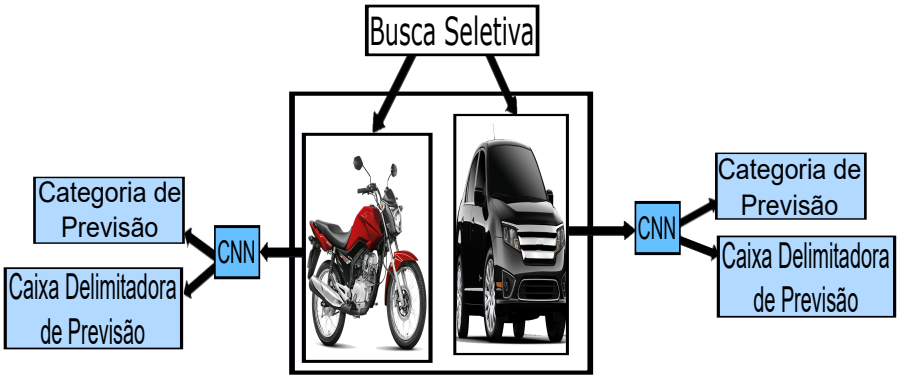
em que $S_{i,j}$ é a esfera de influência de raio $r = 1; x_{i,j}, y_{i,j}, u_{i,j}$ e z são escalares, chamados respectivamente de estado, saída, entrada e limiar da célula $C_{i,j}; a_{k,l}$ e $b_{k,l}$ são pesos escalares. Um *CNN* padrão invariante de espaço com uma vizinhança 3 X 3 é definido uniformemente por uma sequência de 19 números reais, ou seja, **{z, B, A}** com um limite uniforme, um modelo de clonagem feedforward (controle) **B** consistindo em nove pesos de controle $b_{k,l}$, e um modelo de clonagem de feedback **A** consistindo em nove pesos de feedback $a_{k,l}$. Os três termos **{z, B, A}**, chamados de modelo *CNN* (ou gene *CNN* CHUA (1998)), determinam completamente as propriedades dinâmicas do *CNN* . Muitos aplicativos do mundo real podem ser facilmente implementados com um único modelo *CNN* ou uma sequência de modelos *CNN*.

3.1.2.1 R-CNN

O *R-CNN* GIRSHICK et al. (2014), neste algoritmo é realizado a busca de comparação por região e não pixel a pixel como é o caso do algoritmo de *CNN*.

Este modelo escolhe primeiro várias regiões propostas de uma imagem, caixas de âncora é um exemplo deste tipo de método de seleção. Após este procedimento, são rotuladas as caixas delimitadoras e suas categorias, como *offsets*. No próximo processo são realizados avançados cálculos para extrair recursos de cada região escolhida. Em seguida, são utilizado estes recursos das áreas propostas para prever suas caixas delimitadoras e suas categorias (GIRSHICK et al., 2014). A Figura 3.2, ilustra como é o funcionamento do algoritmo do modelo de *R-CNN*.

Figure 3.2 – Algoritmo *R-CNN*



Fonte: Autor (2020)

Estas regiões que foram propostas, são cortadas da imagem e redimensionadas. Em seguida, o algoritmo utiliza CNN para classificar estas regiões que foram recortadas e redimensionadas.

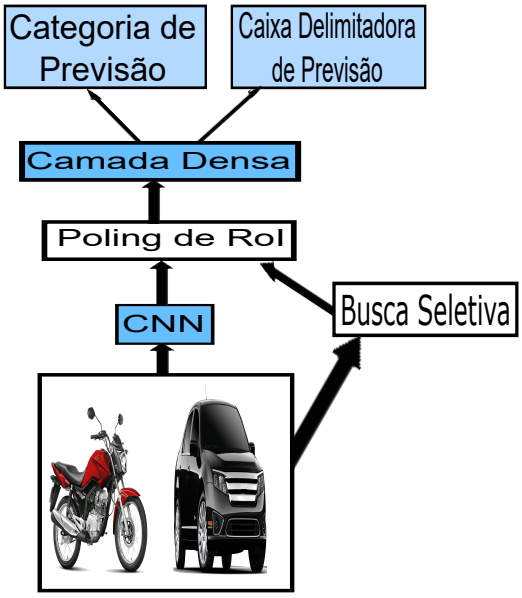
A desvantagem do uso deste algoritmo é a velocidade extremamente lenta (ZHAO et al., 2019). Em uma simples imagem, o *R-CNN* seleciona milhares de áreas propostas partindo de uma única imagem, consequentemente requerendo milhares de cálculos diretos da *CNN* UIJLINGS et al. (2013), devido ao auto custo computacional. Os *R-CNN* não são muito utilizados em aplicações de tempo real, pois necessitam de 40 a 50 segundos para calcular o resultado para várias imagens novas (ZHAO et al., 2019).

3.1.2.2 Fast R-CNN

Logo após o surgimento do *R-CNN*, foi proposto pelo pesquisador *Ross Girshick* junto com a empresa *Microsoft*, o *Fast R-CNN* GIRSHICK (2015). Esse modelo é muito parecido com o *R-CNN*, porém o algoritmo é um pouco diferente.

Devido ao custo computacional do *R-CNN* GIRSHICK et al. (2014) ser muito auto devido aos milhares de cálculos que o algoritmo realiza, sendo muitos deles repetitivos, o *Fast R-CNN* GIRSHICK (2015) surgiu para reduzir estas quantidades de cálculos realizados. O algoritmo realiza apenas cálculos de encaminhamento de *CNN* em toda a imagem (ZHAO et al., 2019). A Figura 3.3 ilustra o funcionamento do algoritmo do *Fast R-CNN*.

Figure 3.3 – Algoritmo *Fast R-CNN*



Fonte: Autor (2020)

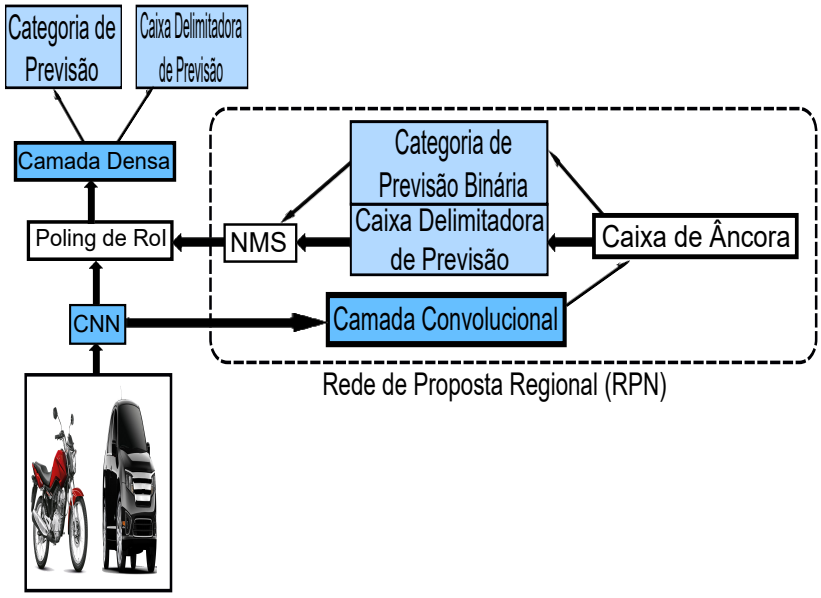
O detector *Fast R-CNN* processa a imagem inteira, ao contrário do *R-CNN* que redimensiona e recorta. O *Fast R-CNN* agrupa os recursos *R-CNN* correspondente de cada região. Os cálculos para regiões sobrepostas são compartilhados, isso faz com que o *Fast R-CNN* seja mais eficiente que o *R-CNN*.

A desvantagem do uso deste algoritmo é que também utiliza o recurso de busca seletiva para detectar regiões de interesse nas imagens (UIJLINGS et al., 2013). Porém, ele utiliza a imagem toda, fazendo com que o processo de detecção seja mais rápido, esse procedimento precisa de quase 2 segundos por imagem, melhorando e muito a velocidade em comparação com o *R-CNN* (ZHAO et al., 2019).

3.1.2.3 Faster R-CNN

O *Faster Region-based Convolutional Neural Network (Faster R-CNN)* REN Kaiming HE e SUN (2016) é uma melhoria do Fast R-CNN (GIRSHICK, 2015). A principal mudança é que a busca seletiva do algoritmo é substituída pelo *Region Proposal Network (RPN)*, pois o Fast R-CNN geralmente demanda de muitas propostas de áreas para a detecção de objetos, o restante do algoritmo continua o mesmo (REN KAIMING HE; SUN, 2016)). Com o uso de *RPN*, o número de áreas propostas gerado é reduzido, com isso a detecção de objetos fica mais precisa. A Figura 3.4, ilustra o funcionamento básico do Faster R-CNN, onde o RPN substitui a busca seletiva do Fast R-CNN.

Figure 3.4 – Algoritmo *Faster R-CNN*



Fonte: Autor (2020)

A desvantagem do uso deste algoritmo é que para tirar os alvos em uma determinada imagem, este sistema precisa de muitas passagens para essas imagem. No entanto, não utilizam a imagens completa, pois, os *RPN* usam apenas partes das imagens (ZHAO et al., 2019).

A Tabela 3.1 mostra a comparação de desempenho dos algoritmos de DL de dois estágios.

Table 3.1 – Comparação de algoritmos de DL de dois estágios

Tipos de algoritmos	Características	Tempo de Detecção	Desvantagens
R-CNN	Este modelo escolhe primeiro várias regiões propostas de uma imagem, caixas de âncora é um exemplo deste tipo de método de seleção, após este procedimento, são rotuladas as caixas delimitadoras e suas categorias, como *offsets*, avançados cálculos para extrair recursos de cada região escolhida. Em seguida, é utilizado estes recursos das áreas propostas para prever suas caixas delimitadoras e suas categorias.	40 a 50 segundos	O tempo necessário para a previsão é grande porque várias regiões passam definitivamente pela CNN e se empregam três modelos distintos para a detecção de alvos.
Fast R-CNN	Para extrair os recursos, cada imagem passa uma vez pela CNN. Todos os modelos distintos aplicados no *R-CNN* são combinados coletivamente para formar um único modelo. Ele emprega um método de pesquisa seletiva nos mapas de recursos para produzir um resultado para o reconhecimento do alvo.	2 segundos	O método usado é prolongado e demorado. Portanto, o tempo de computação ainda é alto.
Faster R-CNN	A abordagem anterior é substituída pelas redes de propostas da região. Portanto, esse procedimento funciona muito mais rápido em comparação com os métodos anteriores.	0.2 segundos	A proposta da região do objeto é demorada. Diferentes categorias de sistemas estão operando em sequência. Isso, o desempenho de todo o procedimento é baseado no funcionamento das operações anteriores.

Fonte: Adaptado de (ADARSH; RATHI; KUMAR, 2020)

3.1.3 Um estágio

Os modelos de algoritmos de dois estágios fornecem uma precisão mais apropriada, porém o tempo para o cálculo é bastante alto. Sendo assim, na literatura visando o tempo de processamento de detecção de objetos, são propostos alguns modelos de algoritmos do DL de um estágio, como é o caso do *SSD* LIU et al. (2016) e o *YOLO* (REDMON et al., 2016). Porém, os primeiros algoritmos deste tipo não alcançaram precisão suficiente ao mesmo tempo, por isso existem algumas variações e melhorias destes algoritmos (ADARSH; RATHI; KUMAR, 2020).

3.1.3.1 You Only Look Once (YOLO)

Na literatura atual, a detecção de objetos é geralmente impulsionada pelo êxito do CNN. O *YOLO* REDMON et al. (2016), utiliza uma abordagem de detecção de objetos unificado e rápida. O processamento dele é básico, utilizando imagens em tempo real a 45 Frames Per Second (FPS) e uma *mean Average Precision (mAP)* de 63.4%, este modelo aprende mais rápido representações gerais dos objetos, superando outros métodos de detecção como o *R-CNN* e o *Faster R-CNN* (REN KAIMING HE; SUN, 2016).

Para obter estes resultados, REDMON et al. (2016) utilizou o conjunto de treinamento *Visual Object Classes 2007 (VOC07)* junto o conjunto de treinamento *Visual Object Classes 2012 (VOC12)*.

O *YOLO*, usa a estrutura do *Darknet* e o conjunto de dados do ImageNet-1000 para treinar o modelo. Porém, o *YOLO*, tem limitações baseadas na proximidade dos objetos na imagem ADARSH, RATHI e KUMAR (2020) e se as proporções do objeto são diferentes das imagens que foram utilizadas nos treinamentos. Esse modelo de algoritmos encontra alguns erros de posição e detecção de objetos e detecta poucos objetos (ZHAO et al., 2019) (JIAO et al., 2019).

A Figura 3.5, mostra a desvantagem do *YOLO*, nela o *YOLO* detecta o homem como um avião.

Figure 3.5 – Falha na detecção de objetos utilizando *YOLO*

Fonte: Adaptado de (ADARSH; RATHI; KUMAR, 2020)

De acordo com TAO et al. (2017), o tempo de execução do *YOLO* é de 0,054 segundos. De acordo com ZHAO et al. (2019), a desvantagem do uso deste algoritmo é a dificuldade para lidar com objetos pequenos em grupos, causados por causa das restrições impostas nas previsões de caixa delimitadora.

3.1.3.2 Single Shot Detector (SSD)

O *SSD* LIU et al. (2016), NING et al. (2017) vem sendo muito utilizado para detecção de objetos. Em MÜLLER e DIETMAYER (2018), os autores propuseram uma solução para identificar e verificar o estado dos semáforos, nesta proposta eles obtiveram a acurácia de até 95%, mesmo com objetos pequenos.

Porém, alguns pesquisadores modificam aos algoritmos do *SSD* melhorando os resultados como a proposta do JEONG, PARK e KWAK (2017), onde obteve um melhoramento na taxa de precisão do *SSD* padrão. O modelo foi adequado para dividir os pesos nas redes de classificação, pelas características, o treino da rede pode ser mais rápido com o melhoramento.

Para o conjunto de treinamento deste modelo foi utilizado o conjunto de treinamento *VOC07* com o *VOC12*, as imagens de entrada com o tamanho de 300 x 300 obteve *mAP* de 78.5%, com 35 de FPS. Já com imagens de 512 x 512 conseguiu *mAP* de 80.8% a 16.6 FPS. A rede proposta obteve resultados de *mAP* melhores que o *SSD* convencional, *YOLO*, *Faster R-CNN* e o *Region-based Fully Convolutional Networks (R-FCN)*.

A Tabela 3.2, mostra resultados dos testes realizados, foi utilizado o *VOC07*, obtendo os seguintes resultados.

Table 3.2 – Resultados da proposta (JEONG; PARK; KWAK, 2017)

Modelo de algoritmos	Tamanho Imagem	*mAP*	FPS
YOLO	448	63.4	45
YOLOv2	416	76.8	67
YOLOv2	544	78.6	40
SSD padrão	300	77.7	61.1
SSD modificado	300	78.5	35.0
SSD padrão	512	79.8	25.2
SSD modificado	512	80.8	16.6

Fonte: Adaptado de (JEONG; PARK; KWAK, 2017)

Existem dois modelos de SSD o SSD300 o qual as imagens de entrada são de 300 x 300, resolução baixa, porém mais rápido e o SSD512, o qual as imagens de entrada são de 512 x 512, apresenta resolução mais alta e mais precisa. A arquitetura do SSD se baseia na arquitetura VGG-16, porem ele descarta as camadas totalmente conectadas.

De acordo com REDMON e FARHADI (2019), o tempo de execução do *SSD* pode chegar a 0,061 segundos. Segundo ZHAO et al. (2019), o *SSD* é mais eficiente e preciso em comparação com o *YOLO*, porém a desvantagem dele é que ele não é capaz de lidar com objetos pequenos.

3.1.3.3 You Only Look Once (Segunda Versão) (YOLOv2)

O *YOLOv2* REDMON e FARHADI (2017) é um modelo de algoritmos que pode identificar e classificar em tempo real cerca de 9000 categorias de objetos. Este modelo de algoritmos pode ser executado com tamanhos variados nas imagens de entrada, possibilitando assim uma troca fácil de velocidade e precisão. A 67 FPS o *YOLOv2* obtém resultados de 76.8% *mAP* no *VOC07*. A 40 FPS, o *mAP* pode chegar a 78.6%, superando o *SSD*. No conjunto de validação do ImageNet, ele obtém 19.7% de *mAP*, já com o COCO ele tem a *mAP* de 16.0%, em todos os conjuntos de validação, os objetos são identificados em tempo real.

Na Tabela 3.3 lista o *mAP* do *YOLOv2* em comparação a outros modelos de algoritmos. No SSD300 foram usadas imagens de 300 x 300, no SSD500 imagens de 500 x 500. Todos os modelos de algoritmos utilizados nos testes usaram o *VOC07*+*VOC12*.

Table 3.3 – Resultados da proposta (REDMON; FARHADI, 2017)

Algoritmos	*mAP* %	FPS
Fast R-CNN	70.0	0.5
Faster R-CNN VGG-16	73.2	7
Faster R-CNN ResNet	76.4	5
YOLO	63.4	45
SSD300	74.3	46
SSD500	76.8	19
YOLOv2 288 x 288	69.0	91
YOLOv2 352 x 352	73.7	81
YOLOv2 416 x 416	76.8	67
YOLOv2 480 x 480	77.8	59
YOLOv2 544 x 544	78.6	40

Fonte: Adaptado de (REDMON; FARHADI, 2017)

O *YOLOv2* adotou uma serie de mudanças para melhor a velocidade e precisão se comparada ao *YOLO*, algumas destas mudanças foram:

A) Normalização de lote: É impraticável normalizar todo o conjunto de treinamento para uma camada *ConvNet*, pois a etapa de otimização usa uma *Stochastic Gradient Descent (SGD)* usa mini lotes durante o treinamento, cada mini lote produz uma estimativa de média e de variação de cada ativação. Sendo que estes elementos de cada mini lote são amostrados da mesma distribuição (ADARSH; RATHI; KUMAR, 2020). Esta operação pode ser vista como uma camada de *Bayesian Network (BN)* (IOFFE; SZEGEDY, 2015). O *YOLOv2*, então coloca uma faixa de *BN* à frente de cada camada convolucional, o que resulta da aceleração da convergência e ajuda a

regular este modelo. A normalização destes lotes atinge uma melhoria de mais de 2% na *mAP* (ADARSH; RATHI; KUMAR, 2020).

B) Classificador de alta resolução: O *YOLO*, adota um classificador com uma resolução 224x244 na entrada e para a detecção utiliza uma resolução 448x448. Este recurso exige que o modelo se adapte a novas entradas. Para solucionar isso, é acrescentado um processo de ajuste fino em 448 a cada 10 épocas no conjunto de dados *ImagiNet*, isso aumenta em ate 4% o *mAP* (ADARSH; RATHI; KUMAR, 2020).

C)Uso de âncoras: Várias âncoras são usadas em um ponto central, para obter várias caixas delimitadoras, onde cada caixa contém 4 parâmetros de coordenadas de posição e 21 informações de probabilidade de categoria (DU et al., 2019).

D) Passo através da camada: O mapa final do *YOLO* é 13 x 13 x 256. Já o *YOLOv2* utiliza o mapa de recursos anterior de 26 x 26 para a detecção de objetos. O mapa recursos 26 x 26 x 256 é amostrado por pontos de linhas e colunas e quatro mapas de recursos de 13 x 13 x 2048. Com isso a taxa para reconhecimento de objetos pequenos é melhorada (DU et al., 2019).

E) Recursos refinados: Com o objetivo de localizar objetos menores, os mapas de recursos de alta resolução podem oferecer informações úteis. O *YOLOv2* concatena estes recursos de alta resolução com os de baixa resolução, acumulando os recursos adjacentes em diferentes canais, o que pode gerar um aumento de 1% no desempenho (ADARSH; RATHI; KUMAR, 2020).

F) Treinamento em várias escalas: Para que este modelo de algoritmos seja robusto ao executar imagens de tamanhos diferentes, este modelo escolhe aleatoriamente uma nova escala de tamanho entre {320, 352, ..., 608} a cada 10 lotes. O que resulta que os algoritmos podem prever objetos em diferentes resoluções. Na alta resolução, o *YOLOv2* atinge 78,6% de *mAP* e 40 FPS em comparação com o *YOLO* com 63,4% de *mAP* e 45 FPS no *VOC07* (ADARSH; RATHI; KUMAR, 2020).

F) Darknet 19: O *YOLOv2* usa arquitetura Darknet 19 que consiste em 19 camadas convolucionais com 5 camadas de pooling máximas e uma camada de função de ativação *Softmax*.

A Tabela 3.4 mostra a arquitetura do *YOLOv2*, darknet-19.

Table 3.4 – Arquitetura darknet-19

Tipo	Filtros	Tamanho / Passo	Saída
Convolucional	32	3 x 3	224 x 224
Maxpool		2 x 2/2	112 x 112
Convolucional	64	3 x 3	112 x 112
Maxpool		2 x 2/2	56 x 56
Convolucional	128	3 x 3	56 x 56
Convolucional	64	1 x 1	56 x 56
Convolucional	128	3 x 3	56 x 56
Maxpool		2 x 2/2	28 x 28
Convolucional	256	3 x 3	28 x 28
Convolucional	128	1 x 1	28 x 28
Convolucional	256	3 x 3	28 x 28
Maxpool		2 x 2/2	14 x 14
Convolucional	512	3 x 3	14 x 14
Convolucional	256	1 x 1	14 x 14
Convolucional	512	3 x 3	14 x 14
Convolucional	256	1 x 1	14 x 14
Convolucional	512	3 x 3	14 x 14
Maxpool		2 x 2/2	7 x 7
Convolucional	1024	3 x 3	7 x 7
Convolucional	512	1 x 1	7 x 7
Convolucional	1024	3 x 3	7 x 7
Convolucional	512	1 x 1	7 x 7
Convolucional	1024	3 x 3	7 x 7
Convolucional	1000	1 x 1	7 x 7
Avgpool		Global	1000
Softmax			

Fonte: Adaptado de (REDMON; FARHADI, 2017)

Conforme WU et al. (2019), o *YOLOv2* teve um tempo de execução de 0,0496 segundos. De acordo com ZOU (2019), várias melhorias foram realizadas no *YOLOv2* em comparação com o *YOLO*, o que resultou em uma melhor precisão de detecção e sua velocidade. No entanto, a desvantagem destes algoritmos é que mesmo com estas melhorias, ainda não se pode comparar a acuraria destes algoritmos com aos de dois estágios.

3.1.3.4 You Only Look Once (Terceira Versão) (YOLOv3)

O *YOLOv3* REDMON e FARHADI (2019), sofreu várias mudanças pequenas na sua configuração, o que resultou em resultados melhores na *mAP* em relação a outros modelos de algoritmos, como o *SSD* ou o *YOLOv2*. Com imagens de 320 x 320, obteve 28.2 *mAP* em 22 ms, tão preciso quando o *SSD* porém muito mais rápido.

Para o teste REDMON e FARHADI (2019) usou-se o conjunto de treinamento COCO, a Tabela 3.5 compara os resultados dos testes realizados por eles, note que além do *YOLOv3* ser mais rápido ele também é mais preciso.

Table 3.5 – Resultados da proposta (REDMON; FARHADI, 2019)

Método	*mAP*	Tempo(ms)
SSD 321	45.4	61
SSD 513	50.4	125
YOLOv3 320	51.5	22
YOLOv3 416	55.3	29
YOLOv3 608	57.9	51

Fonte: Adaptado de (REDMON; FARHADI, 2019)

Segundo DU et al. (2019), o *YOLOv3* melhorou as desvantagens do *YOLOv2*, tornando a velocidade e precisão mais equilibrada. As melhorias foram:

- **Previsões entres escalas:** O *YOLOv3* analisa caixas em escalas diferentes. Ele usa recursos de fusão e amostragem, o que melhora consideravelmente a detecção de alvos pequenos.
- **Previsão de classificação de vários rótulos:** O *YOLOv3* , cada caixa usa uma identificação de vários rótulos para analisar quais classes a caixa delimitadora pode conter. Para tags sobrepostas, a abordagem com vários rótulos pode simular melhor os dados.
- **Extrator de recursos:** No *YOLOv3* para implementar a extração de recursos são usadas 53 camadas sucessivas de 3 x 3 e 1 x 1 onde se chama *Darknet-53*, onde com o *YOLOv2* usa 19 camadas e é denominado *Darknet-19*.

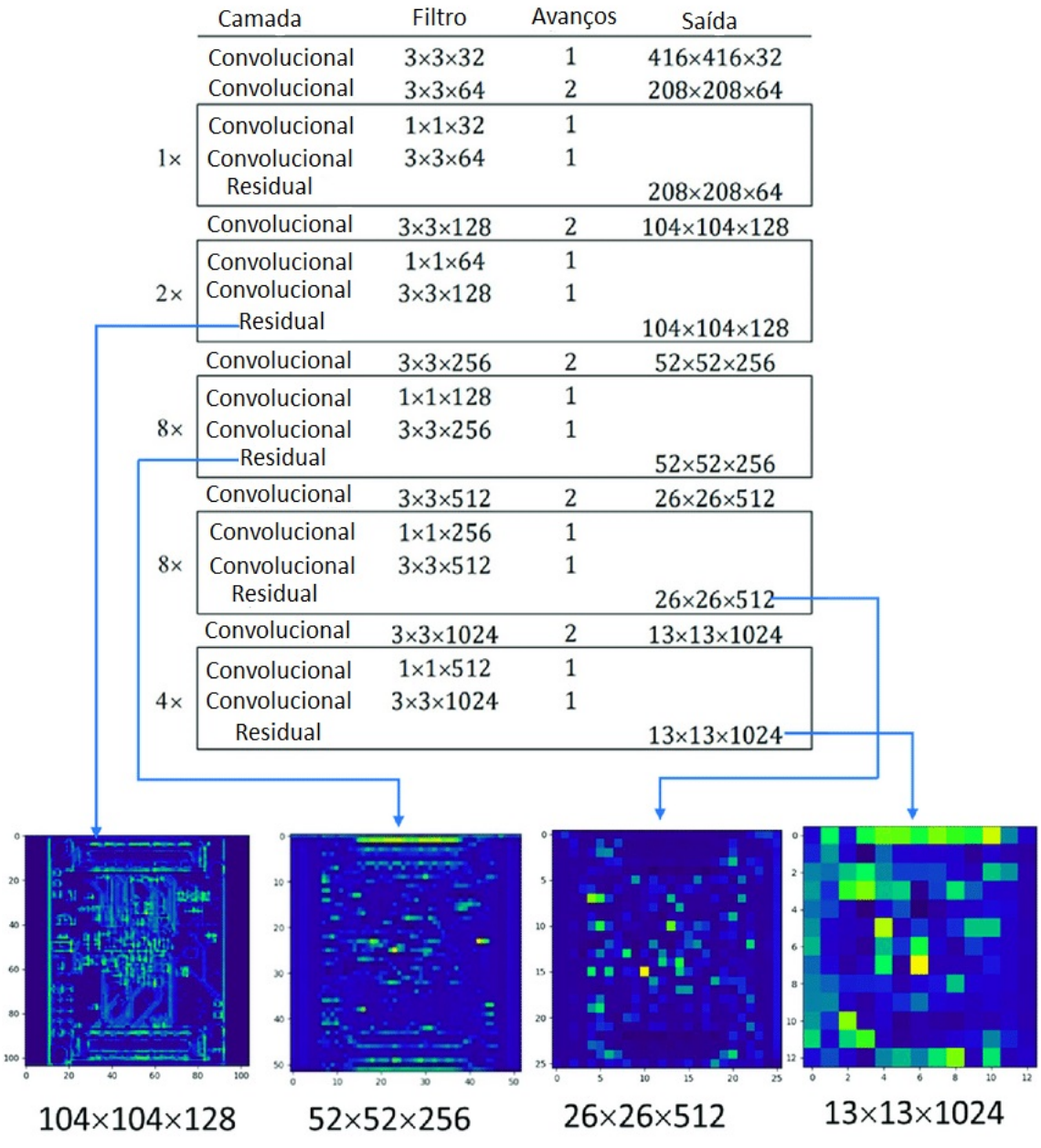
O *Darknet*-53, consiste em velocidade baixa no uso da CPU e uma utilização muito eficiente no uso da GPU (LOU et al., 2019). Este algoritmo é constituído pelas seguintes partes:

- *Região*: As âncoras do parâmetro especificam o valor absoluto da caixa âncora calculado por meios k.
- *Camada de detecção*: Esta parte é baseada em coordenadas e resultados de categoria;
- *A camada YOLO*: Para esta camada precisa especificar muitos parâmetros, como perda de computação, âncoras, etc. O *YOLOv3* usa as três camadas *YOLO* como saída;
- *Camada de amostra superior*: Esta camada realiza 2 vezes a amostragem. O *YOLOv3* usa a camada superior e a camada *YOLO*;

A extração de recurso que *YOLOv3* utiliza é chamada *Darknet-53* que consiste em 53 camadas convolucionais (LOU et al., 2019). Segundo HUANG et al. (2020) o *YOLOv3* usa esta rede para conectar interações de recursos locais e sua função é equivalente à conexão global da camada de recursos completa e à adição de conexões de atalho.

O *YOLO* REDMON et al. (2016) e o *YOLOv2* REDMON e FARHADI (2017) utilizam o *Darknet-19*, que consiste em 19 camadas convolucionais. O *YOLOv3* REDMON e FARHADI (2019) amplia o numero destas camadas para 53. A Figura 3.6, mostra como é a estrutura do *Darknet-53* no *YOLOv3*.

Figure 3.6 – *Darknet-53*



Fonte: Adaptado de (LI et al., 2019)

O funcionamento do *Darknet-53* consiste que estas camadas são como um bloco residual em cada retângulo. Toda a rede constitui em uma cadeia de vários destes blocos com alguns passos entre 2 camadas convolucionais fazendo a redução de sua dimensão. No interior de cada bloco, há uma conexão de ignorar e apenas uma estrutura de gargalo (1x1 seguida por 3x3).

O *YOLOv3* REDMON e FARHADI (2019) foi organizado para detectar objetos em varias escalas, onde também precisa dos recursos de várias escalas. Consequentemente, os três últimos blocos residuais serão todos usados para a detecção posterior. Considerando que a imagem de entrada seja de 416x416, o processo de detecção utilizando a rede do *YOLOv3* REDMON e FARHADI (2019) irá seguir os seguintes passos (HUANG et al., 2020):

1) As imagens de 416x416 são a rede *Darknet-53*. Após executar consequentes convoluções, é obtido um mapa de características de 13x13 e sete vezes por núcleos de convolução 1x1 e 3x3 que são processados realizando a primeira previsão de caixa delimitadora da classe de regressão.

2) Com o mapa de recursos de 13x13 é executado cinco vezes por núcleo de convolução de 1x1 e 3x3, após está operação é processado uma operação de convolução de 1x1, seguido por duas vezes a camada de *upsampling*, obtendo o tamanho de 26x26 no mapa de características. Com este novo mapa de recursos é executado sete vezes usando núcleos de convolução 1x1 e 3x3, obtendo assim a segundo caixa delimitadora.

3) Com este novo mapa de recursos de 26x26. É processado cinco vezes núcleos de convolução de 1x1 e 3x3 , após esta o processamento é processado uma uma operação de dupla amostragem, obtendo mapas de 52x52. Logo após, este mapa é executado sete vezes usando núcleos de convolução de 1x1 e 3x3, obtendo-se assim a terceira categoria delimitadora de regressão.

Segundo HUANG et al. (2020), o *YOLOv3* REDMON e FARHADI (2019) gera três mapas de recursos distintos simultaneamente *13x13*, *26x26* e *52x52*. Cada mapa analisa 3 caixas delimitadoras de regressão, consequentemente cada caixa tem um valor de confiança de destino, 4 coordenadas com seus valores e a probabilidade de arestas diferentes. No caso existem 10647 caixas delimitadoras de regressão, este valor é calculado pelas caixas de regressão (52*52 + 26*26 + 13*13) * 3 = 10647.

Conforme a proposta REDMON e FARHADI (2019), a execução do *YOLOv3* pode chegar a 0,022 segundos, o que torna o algoritmo mais eficiente por parte da execução e alcançando valores consideráveis na acurácia em comparação aos algoritmos de dois estágios.

A Tabela 3.6 mostra a comparação de desempenho alcançados pelos algoritmos de DL de um estágio.

Table 3.6 – Comparação de algoritmos de DL de um estágio

Tipos de Algoritmos	Características	Tempo de execução	Desvantagens
YOLO	O *YOLO* (REDMON et al., 2016), utiliza uma abordagem de detecção de objetos unificado e rápida. O processamento dele é básico, utilizando imagens em tempo real a 45 FPS e uma mAP de 63.4%, com isso o modelo apreende mais rápido representações gerais dos objetos.	0,054 segundos (TAO et al., 2017)	Segundo (ZHAO et al., 2019), a desvantagem do uso deste algoritmo é a dificuldade para lidar com objetos pequenos em grupos, causados por causa das restrições impostas nas previsões de caixa delimitadora.
SSD	Segundo(ZHAO et al., 2019), o *SSD* é mais eficiente e preciso em comparação com o *YOLO*.	0,061 segundos (REDMON; FARHADI, 2019)	A desvantagem é que ele não é capaz de lidar com objetos pequenos (ZHAO et al., 2019).
$YOLO_{V2}$	De acordo com (ZOU, 2019), várias melhorias foram realizadas no *YOLOv2* em comparação com o *YOLO*, o que resultou em uma melhor precisão de detecção e sua velocidade.	0,0496 segundos (WU et al., 2019)	A desvantagem destes algoritmos é que mesmo com estas melhorias, ainda não se pode comparar a acuraria destes algoritmos com aos de dois estágios (ZOU, 2019).
YOLO	Conforme a proposta (REDMON; FARHADI, 2019) a execução do *YOLOv3* pode chegar a 0,022 segundos, o que torna o algoritmo mais eficiente por parte da execução e alcançando valores consideráveis na acurácia em comparação aos algoritmos de dois estágios.	0,022 segundos (REDMON; FARHADI, 2019)	Devido ao seu alto poder de desempenho tanto na acurácia quando no tempo de execução, não foram encontradas desvantagens do uso deste algoritmo em aplicações em tempo real. Pois além do *YOLOv3* original ter alto desempenho, sua estrutura é muito simples, o que facilita a criação ou alterações de novas abordagens de detecção de objetos em tempo real utilizando este tipo de algoritmo.

Fonte: Autor (2020)

3.1.3.5 Soluções baseadas em diferentes versões do algoritmo YOLO

Na literatura, várias propostas de detecção de objetos usam o *YOLO* REDMON et al. (2016) para diferentes aplicações. Essas propostas fazem algumas modificações no algoritmo *YOLO* original para melhorar seu desempenho.

Em TAO et al. (2017), os autores desenvolveram um algoritmo no qual usaram dois modelos de algoritmos, OYOLO que é uma melhoria do *YOLO* no qual foi utilizado para imagens com precisão normal e o OYOLO + *R-FCN* quando a precisão deveria ser mais elevada. Aplicaram dois modelos de validação para o algoritmo em questão, o tempo para identificação dos objetos e a *mAP*. Com este algoritmo proposto, eles obtiveram uma *mAP* de até 69% com *VOC07* junto com o *VOC12*, já com conjunto de treinamento criado por eles, obtiveram resultados de até 83.4%, comparando com outros algoritmos existentes. A Tabela 3.7 mostra a taxa de *mAP* em % do algoritmo proposto.

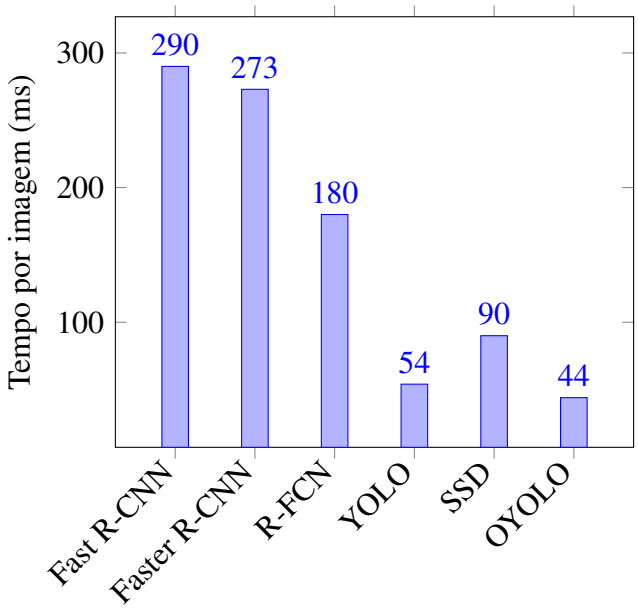
Table 3.7 – Resultados da proposta (TAO et al., 2017)

Modelos de algoritmos	Conjunto de treinamento criado por (TAO et al., 2017)	*mAP* (*VOC07* + VOC12)
Fast R-CNN	-	60.9%
Faster R-CNN	-	62.4%
R-FCN	-	67.7%
YOLO	-	64.0%
SSD	-	64.5%
OYOLO	80.1%	66.0%
OYOLO + *R-FCN*	83.4%	69.3%

Fonte: Adaptado de (TAO et al., 2017)

Com este modelo, TAO et al. (2017) obtiveram resultados satisfatórios com o tempo de identificação dos objetos, demorando apenas 44 ms sendo 10 ms mais rápido que o *YOLO*. A Figura 3.7 mostra a comparação do tempo de identificação em ms perante a outros modelos de algoritmos.

Figure 3.7 – Gráfico tempo para identificação da imagem



Fonte: Adaptado de (TAO et al., 2017)

A câmera infravermelha vem sendo muito utilizada em veículos aéreos não tripulados devido à facilidade de adaptação e a baixa luminosidade (ZHANG; ZHU, 2019). Entretanto a sua baixa qualidade na captação destas imagens fica difícil implementar uma detecção de veículos utilizando estas imagens.

Em ZHANG e ZHU (2019) os autores propuseram um método *YOLOv3* modificado, no qual foi construído uma estrutura com apenas 16 camadas, expandiram as caixas de ancoragem para quatro escalas, assim melhorando a detecção de veículos menores. A Tabela 3.8, mostra os resultados, da precisão, do recall, do F1-Score, estes dados são chamados de métricas de validação onde serão explicados na seção 4.2. Para os testes foram utilizadas a base de dados VIVID e o NPU:

- VIVID - É um conjunto de imagens aéreas públicas, onde é composto por oito imagens, sendo cinco sequências de imagens visíveis e três sequências de imagens infravermelhas. Devido que as sequências de imagens são bastante parecidas, ZHANG e ZHU (2019) foram selecionadas 400 imagens visíveis para uma rede *YOLOv3* pré-treinada, e 100 imagens de infravermelhas para o aprendizado.

- NPU - As imagens deste conjunto foram capturadas em Xi'an, China. As imagens foram capturadas da mesma maneira que as imagens do VIVID. Foram selecionadas 268 imagens de aéreas infravermelhas.

Table 3.8 – Resultados da proposta (ZHANG; ZHU, 2019)

Conjunto de dados	Imagens	Veículos	TP	FP	FN	Precisão	Recall	F1-Score
VIVID_teste_01	112	425	398	24	8	94.3%	98.0%	96.1%
VIVID_teste_02	163	586	569	30	15	95.0%	97.4%	96.2%
NPU	170	616	587	16	18	97.4%	97.0%	97.2%
Total	445	1627	1554	70	41	95.7%	97.4%	96.5%

Fonte: Adaptado de (ZHANG; ZHU, 2019)

Apesar que várias cidades monitorarem suas vias públicas com um monitoramento de câmeras visando o melhoramento do tráfego, ainda existem várias inspeções manuais uma delas é a verificação de estacionamento em local proibido. Visando a melhoria deste processo CHEN e YEO (2019), propôs a automatização deste processo usando o *YOLOv3* para detectar veículos. Nos testes realizados obtiveram resultados com uma precisão de acerto muito alto apesar das posições das câmeras, iluminação do vídeo e condições climáticas. Para o treinamento da rede CHEN e YEO (2019), utilizou o conjunto de dados COCO, pois contém apenas 80 classes, com isso eles diminuíram a taxa de falsos positivos e aumentaram a *mAP*.

Detectar rostos é uma das tarefas mais básicas em aplicações de fisionomia. Porém, a detecção de rostos pequenos em imagens é um problema muito complexo e desafiador para os modelos de detecção de padrões, YANG et al. (2019) propôs combinar o *You Only Look Once (tiny Version 3) (YOLOv3-tiny)* com um método de regularização usando na camada convolucional, na qual as unidades de ativação estão espacialmente inter-relacionadas *(Dropblock)*. Após as experiências realizadas mostraram que o *YOLOv3-tiny* combinado com Dropblock, obteve um resultado de 7.74% na taxa de precisão em comparação a modelos de *YOLOv3-tiny*. Para realizar os testes utilizaram dois conjuntos de dados o Face Detection Data Set and Benchmark (FDDB) e o WiderFace:

- FDDB - É muito usado para pesquisas de detecção de rosto. Este conjunto conta com 2845 imagens tiradas no ambiente de campo e o número total de faces pode chegar a 5171, destas imagens foram selecionadas 900 de modo randômico para o conjunto de testes.

- WiderFace - É uma base de dados proveniente da Universidade Chinesa de Hong Kong. Ela contém 32203 imagens, incluindo 393703 rotos. Foram selecionadas aleatoriamente 6900 imagens, dentre as quais 5400 foram escolhidas como conjunto de treinamento, 600 como conjunto de validação e 900 conjuntos de testes.

A Tabela 3.9, mostra o resultado dos testes realizados.

Table 3.9 – Resultados da proposta (YANG et al., 2019)

Método	Base de Dados	*mAP %*
YOLOv3-tiny com Dropblock	FDDB	87.96
YOLOv3-tiny com Dropblock	Wilder-Face	46.59
YOLOv3-tiny	FDDB	87.14
YOLOv3-tiny	Wilder-Face	38.85

Fonte: Adaptado de (YANG et al., 2019)

3.1.3.6 Soluções no tráfego urbano utilizando diferentes versões do algoritmo YOLO

Visando à segurança pública PENG et al. (2016), propôs a detecção de pedestres usando o *YOLO* com base no *Gaussian Mixture Model* (GMM) ou modelo de mistura gaussiana, onde foi utilizado para modelar o plano de fundo e detectar pedestres com antecedência. O modelo GMM foi utilizado para subtração do fundo das imagens, ele modela o valor de cada pixel único, pois é uma função paramétrica de densidade de probabilidade representada com uma soma ponderada das densidades dos componentes gaussianos.

A proposta REN, FANG e DJAHEL (2017) consiste em uma contagem de pessoas por blocos em tempo real. Nesta proposta, os autores apresentam o YOLO-PC, que tornou a contagem de pedestres mais direcional com resultados mais precisos e rápidos. Além de ser um algoritmo que é capaz de reconhecer pessoas irrelevantes e ignorá-las no processo de contagem.

Utilizando o *YOLOv2* REDMON e FARHADI (2017), durante o processo de detecção, poderá haver perda de informações sobre pedestres, o que causará o desaparecimento de gradientes, causando a detecção imprecisa de pedestres LAN et al. (2018), propôs o YOLO-R, que é uma melhoria na estrutura do *YOLOv2*, este modelo foi testado utilizando um conjunto de dados de pedestres do *INRIA*. A Tabela 3.10 mostra a taxa de precisão e recall do YOLO-R em comparação com o *YOLOv2*.

Table 3.10 – Resultados da proposta (LAN et al., 2018)

	YOLOv2	YOLO-R
Precisão	97.37%	98.56%
Recall	89.33%	91.21%

Fonte: Adaptado de (LAN et al., 2018)

Detecção de pedestres é uma das aplicações a qual foi vista como essencial para o mundo real, visando isso, VALIATI e MENOTTI (2019) propôs detectar pedestres, usando

o *YOLOv3*, utilizando uma técnica inspirada no *Simple Denoising and Super-Resolution Convolutional Neural Network (SD-SRCNN)*, fase de infusão de segmentação semântica fraca, o que aumentou a sua precisão sem afetar na velocidade de inferência.

Erros e omissões na detecção de pedestres, geralmente são constantes em detecção de objetos, em modelos de algoritmos de um estágio; porém ZHAO e CHEN (2019), propôs um método aprimorado de detecção baseado no *YOLOv3*, o qual melhora o campo de detalhes da análise humana e a área de saliência em tempo real para detectar pedestres. Nessa proposta, eles abordaram duas questões, que foram:

- Extrair mais atributos distintos com a investigação humana.
- Impulsionar mais a detecção da região de saliência em tempo real através de câmeras de vigilâncias.

A detecção de semáforos de pedestres usando uma câmera de celular enfrenta muitos desafios. Os semáforos para pedestres têm aparências diferentes em diferentes países e até para diferentes fabricantes, a distância para um semáforo de pedestres pode variar (geralmente de 4 a 20 metros) e, portanto, a escala de um semáforo pode variar. Eles podem ser temporariamente obstruídos por veículos e outros pedestres, ademais pela iluminação varia dependendo da hora do dia e do clima (ROTHAUS; ROTERS; JIANG, 2009).

Visando a esses desafios ASH et al. (2018), propõem-se um método de detecção destes semáforos baseado no *You Only Look Once (tiny Version 2) (YOLOv2-tiny)*. O conjunto de dados contém 950 imagens coloridas - 450 luzes verdes 'Andar' e 500 luzes vermelhas 'Não andar'. Nos testes apresentados por eles foi demonstrado que o método foi mais preciso que os modelos de dois estágios. A Tabela 3.11, mostra a precisão, o recall e o tempo de processamento para a detecção do objeto.

Table 3.11 – Resultados da proposta (ASH et al., 2018)

	Andar		Não Andar		Tempo de
	Precisão	Recall	Precisão	Recall	processamento em (ms)
Faster R-CNN	98.8	94.2	98.1	97.6	50
YOLOv2	100	98.31	100	97.14	15.4
YOLOv2-tiny	100	94.92	100	100	6

Fonte: Adaptado de (ASH et al., 2018)

O trânsito de grandes centros urbanos são bastante complexos e mutáveis, principalmente nos cruzamentos e a detecção de policias de trânsito sendo parte crucial para veículos

autônomos. Em vista disso ZHENG et al. (2018) propôs um método de detecção de policias de trânsito. Nesse modelo eles alcançaram 77% de *mAP*, a velocidade de detecção atingiu 45 FPS. Eles utilizaram a base de dados do ImageNet para os testes.

Na proposta LIN e SUN (2018), utilizam o *YOLO* para a contagem de veículos em tempo real. Durante os testes, eles utilizaram imagens de trânsito no período da manhã, tarde e noite onde obterão resultados de contagem satisfatórios.

Para estimar a densidade e a complexidade do tráfego urbano a contagem de veículos é uma das partes principais para estes processos. Sistemas de administração de tráfego urbano, em tempo real, tornou-se muito popular na literatura, devido ao número elevado de câmeras de monitoramento do trânsito e análise de *big data*.

Com isso ASHA e NARASIMHADHAN (2018), propôs um método de contagem de veículos baseado em vídeos do tráfego urbano, estes vídeos foram capturados usando câmeras portáteis. Este modelo foi divido em três etapas, utilizando o *YOLO* para a detecção dos veículos, com filtros de correlação e contagem. A Tabela 3.12, mostra a precisão, *recall*, *F1-Score* e a contagem da acurácia do (ASHA; NARASIMHADHAN, 2018).

Table 3.12 – Resultados da proposta (ASHA; NARASIMHADHAN, 2018)

Video	Número de frames	Precisão %	Recall %	F1-Score %	Acurácia %
1	899	100	100	100	100
2	715	93.7	100	96.7	93.7
3	845	100	94.4	97.1	94.4
4	3598	94.8	100	97.3	94.8
5	2100	92	100	95.8	92.0
6	5528	98.5	98.5	98.5	97.0
7	2999	100	100	100	100

Fonte: Adaptado de (ASHA; NARASIMHADHAN, 2018)

O modelo de algoritmos do DL de um estágio, como o *YOLO* vem sendo muito utilizado em aplicação para o ambiente real pois possui requisitos mais baixos se comparado com os outras algoritmos de dois estágios como o *Faster R-CNN* (REN KAIMING HE; SUN, 2016).

Ressaltando isso, DU et al. (2019) propôs um método de detecção em tempo real de veículos e semáforos, usando o *YOLOv3*, para o conjunto de dados eles expandiram o conjunto de dados do apollo Huang et al. (2018), no qual continha apenas 200 imagens onde criaram um novo conjunto de dados que chamaram de veículo e semáforo (V-TLD). Este conjunto de dados contém imagens de 2018, em ambientes urbanos em *Wuhan*, China. O período da coleta foi

entre as 8h00 às 17h00, o tamanho das imagens tem 1920 x 1080. Este conjunto de dados foi divido em 1348 imagens para treinamento e 670 imagens para testes.

A Tabela 3.13, mostra a precisão, o recall e o tempo em segundos.

Table 3.13 – Resultados da proposta (HUANG et al., 2018)

Modelo	Avaliações	Veículo	Semáforo	Tempo (segundos)
YOLOv2	Precisão	96.16 %	95.59 %	0.045
	Recall	72.34 %	75.38 %	

Fonte: Adaptado de (HUANG et al., 2018)

Em DASGUPTA, BANDYOPADHYAY e CHATTERJI (2019), propôs um método de detecção de infrações utilizando dois métodos de detecção de objetos, o *YOLOv3* e algoritmos baseados em *CNN*. Nesse o *YOLOv3* foi utilizado para detectar motociclistas e o *CNN* foi utilizado para verificar se ele esta usando ou não o capacete. O modelo alcançou a taxa de 96,23% na taxa de precisão para detecção do capacete.

Em LAM, NG e CHAN (2019), propõe-se um método de detecção de veículos utilizando o *YOLOv3* , as taxas alcançaram o valor de 86% de precisão e a taxa de recuperação chegou a 87%, dependendo da localização e ângulo destas câmeras.

Detectar e rastrear veículos no trânsito são atividades fundamentais em técnicas de tráfego urbano. Com isso LOU et al. (2019) propõe um sistema que é capaz de realizar a detecção e o rastreamento de veículos utilizando o *YOLOv3* junto com o filtro de *Kalman*. Nos testes atingiu um *mAP* de 92,11% na contagem de veículos em vias congestionadas a uma velocidade de 2,55 FPS. A Tabela 3.14, mostra a comparação da acurácia do algoritmo tradicional com o algoritmo proposto por eles.

Table 3.14 – Resultados da proposta (LOU et al., 2019)

Ambiente Experimental	Número real de veículos	Algoritmo Tradicional		Algoritmo Proposto	
		Número de rastreamento de contagem	Acurácia %	Número de rastreamento de contagem	Acurácia %
1	22	31	56	20	90.9
2	32	36	87	31	97
3	352	-	-	323	91.76

Fonte: Adaptado de (LOU et al., 2019)

A solução CHEN e LIN (2019) propõe a detecção de veículos em tempo real utilizando YOLOv3-tiny que é um aprimoramento do *YOLOv3-tiny*. Durante os testes eles obterão 87,79% de *mAP* e a velocidade de detecção atinge 28 FPS. A Tabela 3.15 mostra a comparação de precisão, *recall* e *F1-Score* entre *YOLOv3-tiny* e o YOLOv3.

Table 3.15 – Resultados da proposta (CHEN; LIN, 2019)

	Precisão	Recall	F1-Score
YOLOv3-tiny	88%	81%	84%
$YOLO_{V3} - look$	91%	82%	86%

Fonte: Adaptado de (CHEN; LIN, 2019)

Segundo HUANG et al. (2017), algoritmos de detecção de objetos de dois estágios baseados em *CNN* como *R-CNN* GIRSHICK et al. (2014), *Fast R-CNN* GIRSHICK (2015) e o *Faster R-CNN* REN Kaiming HE e SUN (2016) são modelos muito precisos, porém a velocidade de detecção é bastante demorada, o que não é muito viável para aplicações em tempo real.

Porém modelos de detecção de um estágio como o *SSD* LIU et al. (2016) e o *YOLO* REDMON et al. (2016) são mais rápidos para detectar objetos nas imagens, mas com um taxa de precisão relativamente baixa se comparada aos modelos de dois estágios.

Na literatura existem várias propostas de melhoria destes algoritmos de um estágio como o *YOLOv3* REDMON e FARHADI (2019), onde existem melhorias, almejando sempre na taxa de precisão. Além disso, em algumas, pode-se até superar as taxas de precisão de algoritmos de dois estágios, como é o caso do *YOLOv3-tiny* (GONG et al., 2019).

É muito difícil determinar qual o melhor sistema de detecção de objetos atual ADARSH, RATHI e KUMAR (2020), pois cada um tem sua aplicação e particularidade perante a problemas reais. Aplicações onde o foco é a precisão e não o tempo de detecção é recomendado usar algoritmos de dois estágios baseados em CNN, já em aplicações em tempo real é recomendado usar algoritmos de um estágio como o *SSD* e o *YOLO*, devido a sua velocidade de processamento.

3.1.3.7 Soluções de semáforos usando Inteligência Artificial

A solução proposta por LEE e CHUNG (2017), teve uma acurácia de 90,24. Nesta proposta eles utilizaram três técnicas de redes neurais convolucionais que foram: *AlexNet* RUSSAKOVSKY et al. (2014), *GoogLeNet* SZEGEDY et al. (2015) e *ResNet18* (HE et al., 2016). A Tabela 3.16, apresenta as taxas de acertos em % do trabalho realizado por LEE e CHUNG (2017), nela a taxa de precisão de acerto chegou a 95.07% na detecção de picapes.

Table 3.16 – Resultados da proposta de (LEE; CHUNG, 2017)

	Taxa de acertos %									
	Caminhão Articulado	**Bicicleta**	**Ônibus**	**Carro**	**Moto**	**Veículo não Motorizado**	**Pedestres**	**Picape**	**Caminhão Pequeno**	**Van de Empresas**
Caminhão Articulado	93.58	0.00	0.19	0.50	0.00	0.62	0.00	0.15	4.72	0.08
Bicicleta	0.00	87.74	0.70	0.50	4.20	0.18	6.65	0.00	0.00	0.00
Ônibus	0.35	0.00	96.20	0.43	0.00	0.58	0.00	0.12	0.58	1.59
Carro	0.01	0.00	0.00	98.89	0.00	0.01	0.00	0.88	0.01	0.15
Moto	0.00	0.61	0.00	5.66	92.12	0.40	0.40	0.81	0.00	0.00
Veículo não Motorizado	4.79	0.23	0.23	9.13	0.46	68.72	0.46	2.05	10.27	1.60
Pedestres	0.00	2.68	0.00	1.02	0.83	0.13	94.25	0.00	0.00	0.00
Picape	0.00	0.00	0.03	4.56	0.00	0.06	0.00	95.07	0.18	0.08
Caminhão Pequeno	7.73	0.00	0.23	2.11	0.00	0.63	0.00	4.53	82.89	1.56
Van de Empresas	0.00	0.00	0.21	14.00	0.00	0.04	0.00	1.57	0.66	83.53
Nenhum Veículo	0.01	0.00	0.01	0.28	0.00	0.02	0.02	0.00	0.00	0.00

Fonte: Adaptado de (LEE; CHUNG, 2017)

Algumas propostas de semáforos inteligentes usam lógica difusa ou lógica *fuzzy* como o caso do MOGHADDAM, HOSSEINI e SAFABAKHSH (2015), em que a lógica *fuzzy* é usada para contar o número de veículos que passam pelo cruzamento, mas usa sensores para contabilizar os veículos. Já ZAID, SUHWEIL e YAMAN (2017), propôs esse mesmo conceito, mas, em vez de sensores, usou imagens para determinar o tempo de espera de cada veículo no cruzamento. No entanto, Fujitsu LIU, LIU e CHEN (2017), desenvolveu um projeto audacioso, onde ele usa a Internet das coisas com o uso de câmeras para controlar semáforos em uma cidade; no entanto, esse projeto foi implementado usando um simulador.

Outra proposta de semáforo que utilizou simulação foi a HARAHAP et al. (2019), na qual eles propuseram um menor tempo de espera para filas nos cruzamentos, com base na entrada de veículos nesses cruzamentos. No entanto KANUNGO, SHARMA e SINGLA (2014) propôs um método de maneira semelhante, no entanto, eles usaram imagens de vídeo em tempo real das câmeras nos cruzamentos e, com base na densidade do tráfego, alteram os semáforos através de um algoritmo, a fim de reduzir o congestionamento do tráfego.

Na solução de TAHMID e HOSSAIN (2017), eles propuseram um sistema para controlar a densidade do tráfego em tempo real usando o processamento de imagens digitais, obtendo uma melhor eficiência em geral aos sistemas existentes. Para evitar paradas de veículos em cruzamentos sob condições de tráfego baixo, SILVA, AQUINO e MEIRA (2015), propõem um semáforo inteligente que detecta a presença de veículos na estrada usando vários dispositivos de entrada, como radares, câmeras ou sensores.

Também, existem propostas para semáforos que usam um sistema IoT com *Passive Infrared Sensor (PIR)*, conhecido popularmente como sensores de presença e o *Raspberry Pi* (DÍAZ; GUERRA; NICOLA, 2018). Nesta proposta, eles criaram um semáforo totalmente automatizado que altera o tempo de espera no cruzamento automaticamente. Enquanto isso, NELLORE e HANCKE (2016) propôs um semáforo com o auxílio de sensores sonoros e imagens de trânsito para dar prioridade aos veículos de segurança e saúde pública, mas as imagens capturadas são de baixa precisão. Este semáforo pode gerar um custo mais alto devido ao custo desses sensores.

O presente trabalho usará o algoritmo *YOLOv3*. Como mencionado anteriormente. Esse algoritmo tem demonstrado alto poder na velocidade de detecção de objetos na literatura, e mantendo elevado número na taxa de *mAP%*, além de estar bastante consolidada em trabalhos com processamento de imagens para identificação e classificação de objetos.

Atualmente existe uma proposta de semáforo que dá prioridade para veículos de socorro, porém utilizam sensores sonoros e imagens, mas com baixa precisão NELLORE e HANCKE (2016). Este tipo de semáforo pode gerar um custo maior devido ao custo destes sensores.

A Tabela 3.17, mostra algumas propostas de reconhecimento de veículos e de semáforos inteligentes e a proposta do presente trabalho com uma nova abordagem.

Table 3.17 – Trabalhos Relacionados

Pesquisadores	Pesquisa	Semáforo	Sensores	Rede Neural	*fuzzy*	DL	Acurácia	Contribuição
(WU; TSAI, 2016)	Pedestrian, bike, motorcycle, and vehicle classification via deep learning: Deep belief network and small training set	Não	Não	Sim	Não	Sim	89,5%	Detecção de bicicletas, Veículos, Motos e Pedestres com acurácia alta
(LEE; CHUNG, 2017)	Deep Learning-Based Vehicle Classification Using an Ensemble of Local Expert and Global Networks	Não	Não	Sim	Não	Sim	92,98%	Detecção de bicicletas, Veículos, Motos e Pedestres com acurácia alta
(ZAID; SUHWEIL; YAMAN, 2017)	Smart controlling for traffic light time	Sim	Não	Sim	Sim	-	-	Tempo de espera no cruzamento reduzido
(MÜLLER; DIET-MAYER, 2018)	Detecting Traffic Lights by Single Short Detection	Não	Não	Sim	Não	Sim (*SSD*)	95,0%	Detecção mais precisa de Placas e semáforos
(NELLORE; HANCKE, 2016)	Traffic Management for Emergency Vehicle Priority Based on Visual Sensing	Sim	Sim	Sim	Não	Não	-	Prioridade para veículos de socorro usando sensores sonoros e imagens
Barbosa, C. Rodrigo 2020	Classificação de veículos baseada em Deep Learning para uso em semáforos inteligentes	Sim	Não	Sim	Não	Sim (*YOLOv3* modificado *PVInet*	98,0%	Classificação e Detecção de veículos públicos como Ambulância, Bombeiros, Carro, Ônibus e Polícia com alta acurácia. Dar prioridade para estes veículos em questão em um semáforo inteligente.

Fonte: Autor (2020)

4 METODOLOGIA

Neste capitulo, terá uma descrição mais completa das principais etapas da solução de semáforo inteligente proposto. A Figura 4.1, ilustra como foi desenvolvido o trabalho. O fluxograma geral é composto por 4 partes. Que são:

A) Base de dados

É um conjunto de dados formado por imagens homogêneas para o treinamento do modelo do DL.

B) Treinamento *(Deep Learning)*

O modelo escolhido foi o *PVInet* que é uma melhoria do $YOLO_{V3}$, ele obteve uma alta taxa de precisão e um tempo de resposta muito rápido.

C) Validação *Deep Learning*

Após o treinamento do DL usando o *PVInet*, foi realizado uma fase de testes utilizando imagens da BD para averiguar se o modelo utilizado realmente está realizando detecção de veículos.

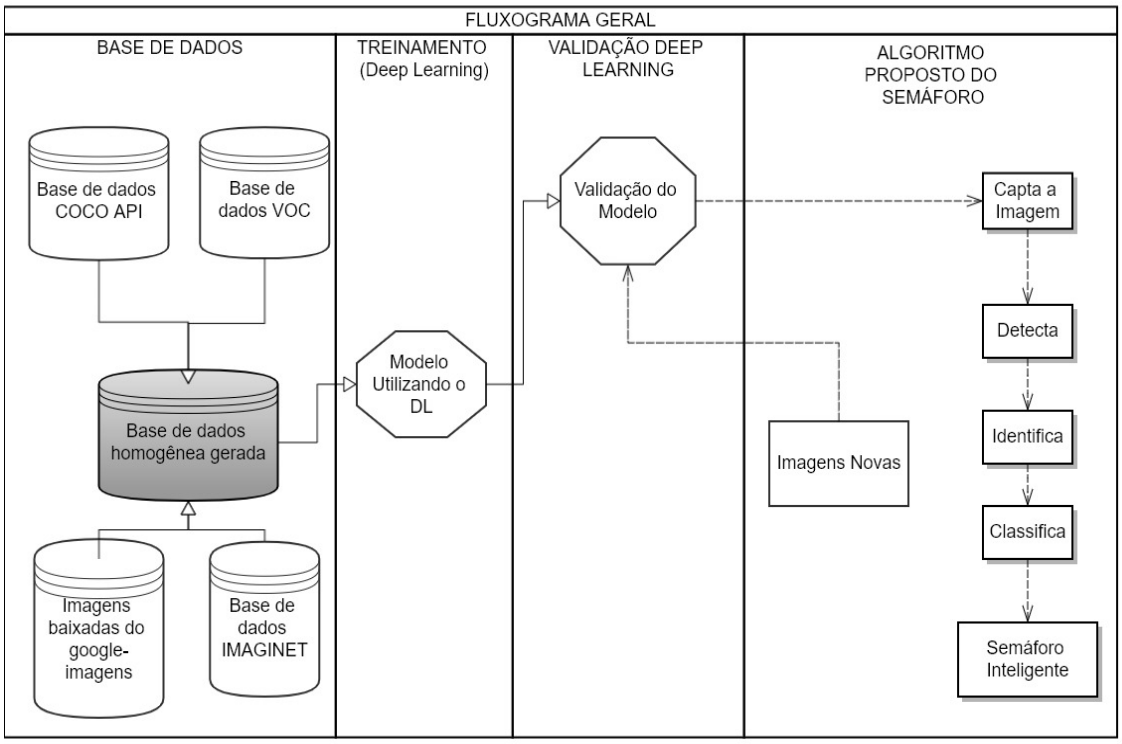
D) Algoritmo Proposto do Semáforo

O algoritmo do DL funcionará da seguinte forma:

1. Captação de imagens em tempo real
2. Detecção de veículos na via
3. Identificação de veículos
4. Classificação de veículos
5. Após o algoritmo do DL classificar os veículos, serão enviados sinais para a liberação ou não da via para o semáforo inteligente, de acordo com sua prioridade.

Vale ressaltar que o algoritmo do semáforo foi uma proposta teórica a qual não foi implementada ainda, mas poderá ser em trabalhos futuros. Após a implementação do modelo do DL usando o *PVInet* imagens em tempo real serão utilizadas, no caso estas imagens são representadas em Imagens novas.

Figure 4.1 – Fluxograma geral da Metodologia



Fonte: Autor (2020)

4.1 Base de dados

Devido a dificuldade de encontrar uma BD homogênea para treinamento do DL, foi construída uma nova BD para realizar treinamento e testes para a solução proposta. Foram utilizados quatro BD distintas, que são: *image-net* [1], *voc* [2], *coco-api* [3] e o *google* imagens (imagens buscadas diretamente na internet)[4].

A BD é composta por 5 classes de imagens (Ambulância, Bombeiro, Carro, Ônibus e Polícia) e subdividida em 3 subcategorias Direita (Di), Esquerda (Es) e Frontal (Fr). O total de imagens deste BD é composta por 4650 imagens coloridas distintas que são subdivididas em 310 imagens para a direita, 310 para a esquerda e 310 imagens de frente totalizando 930 imagens para cada classe. A resolução de cada imagem foi alterada para 1280 x 720.

[1] Disponível em <http://image-net.org/>
[2] Disponível em <http://host.robots.ox.ac.uk/pascal/VOC/>
[3] Disponível em <http://cocodataset.org/#home>
[4] Disponível em <https://www.google.com/imghp?hl=pt-br>

A Figura 4.2 mostra a classe de ambulâncias, a Figura 4.3 mostra a classe de caminhão de bombeiros, a Figura 4.4 mostra a classe de carro, a Figura 4.5 mostra a classe de ônibus (transporte publico urbano) e a Figura 4.6 mostra a classe de polícia.

A) Classe Ambulância

Figure 4.2 – Imagens da BD - CLASSE AMBULÂNCIA

Fonte: Imagens do banco de dados criado (2020)

B) Classe Bombeiro

Figure 4.3 – Imagens da BD - BOMBEIRO

Fonte: Imagens do banco de dados criado (2020)

C) Classe Carro

Figure 4.4 – Imagens da BD - CLASSE CARRO

Fonte: Imagens do banco de dados criado (2020)

D) Classe Ônibus

Figure 4.5 – Imagens da BD - CLASSE ÔNIBUS

Fonte: Imagens do banco de dados criado (2020)

E) Classe Polícia

Figure 4.6 – Imagens da BD - CLASSE POLÍCIA

Fonte: Imagens do banco de dados criado (2020)

Após a criação da BD, é necessário selecionar a região de interesse para que a rede possa ser treinada. A ferramenta utilizada foi o *Image Labeler* a qual pertence ao *Matlab2019*, para usar esta ferramenta é necessário instalar o *resnet50*, que é um pacote de rede neural convolucional do *Matlab2019*. Com esse pacote, é possível realizar o treinamento da rede. A Figura 4.7, ilustra algumas imagens da BD onde foi utilizado o *Image Labeler*.

Figure 4.7 – Imagens da BD com *Image Labeler*

(a) Figura 1 (b) Figura 2

(c) Figura 3 (d) Figura 4

Fonte: Autor (2020)

As coordenadas das regiões de interesse das imagens apresentadas na Figura 4.7, são apresentadas na Tabela 4.1. Após o uso do *Labeling*, as coordenadas das imagens são as seguintes:

Table 4.1 – Coordenadas das Figuras 4.7

Figura	Coordenadas
a	47,41,331,361
b	47,41,331,361
c	13,9,926,524
d	1,18,282,151

Fonte: Autor (2020)

4.1.1 Modelo da rede PVInet

Como mencionado anteriormente, neste trabalho, é proposta uma estratégia de concatenação de recursos, na qual os mapas de recursos aprendidos com cada bloco de imagem são concatenados com todos os blocos subsequentes. Esses blocos são usados como entrada por

meio de *pool*. Assim, os mapas de recursos de todas as saídas do bloco são concatenados, na rede *backbone*, como entrada para o módulo de detecção. Por meio da reutilização e propagação de recursos, os mapas de recursos de entrada apresentam um poder de representação enriquecido. O mapa fornece informações adicionais para aprendizagem de características, que pode ser definida pela equação 4.1.

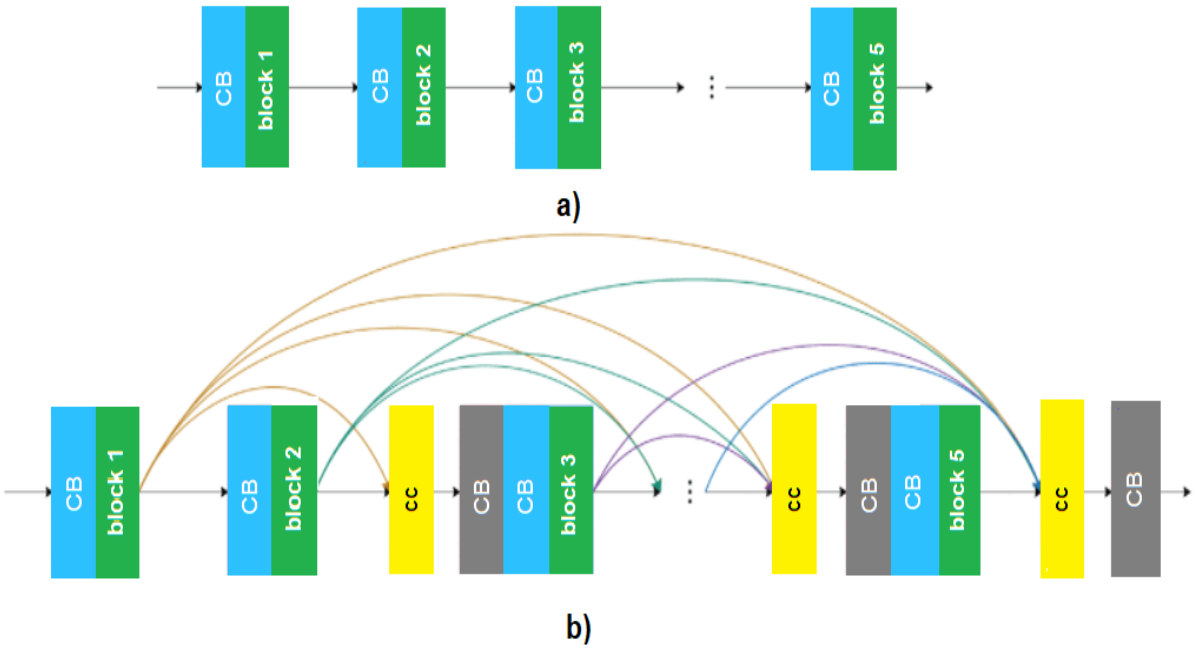
$$VehIBlock_i = Conc\left[MaxP\left(VehOBlock_1, VehOBlock_2, ..., VehOBlock_{i-1}\right)\right] \tag{4.1}$$

onde *VehIBlocki* e *VehOBlocki* representam os mapas de recursos de entrada e saída na rede de *backbone*, respectivamente. A variável *MaxP* representa a operação de *pool* máximo; a variável *Conc* representa a operação de concatenação.

Uma camada de normalização em lote e o *SRS* são usados na rede. Eles são usados para redução de dimensão e convergência acelerada. O *SRS* pode ajustar a saída através de um par de parâmetros treináveis independentes, apresentando melhor desempenho de generalização e maior velocidade de aprendizado. Nos experimentos, o *Softmax* e o *ReLU* foram testados para comparação.

O modelo $YOLO_{V3}$ original apresenta vários blocos residuais e este fato traz um grande número de parâmetros para a rede. Muitos parâmetros levam a um treinamento de tempo prolongado e diminuem a velocidade de detecção do modelo. Assim, a estrutura do *PVInet* precisa ser otimizada para trabalho em tempo real. No *PVInet*, uma modificação no $YOLO_{V3}$ foi realizada, conforme é mostrado na Figura 4.8.

Figure 4.8 – *Backbone* $YOLO_{V3}$ original x *Backbone* do *PVInet*



(a)Representação da estrutura do *backbone* do $YOLO_{V3}$ original;

(b)Representação da estrutura do *backbone* do *PVInet*.

Fonte: Autor (2020)

A solução *Dense Block* HUANG et al. (2017) apresenta algumas vantagens, como eficiência computacional e de armazenamento. Por este motivo, será utilizado no *PVInet*. O *Densely Connected Convolutional Networks (DenseNet)* necessita apenas da metade dos parâmetros da rede, para a mesma precisão de predição, diminuindo a complexidade do modelo e agilizando a detecção dos veículos.

A estrutura de *Dense Connection* das camadas convolucionais, que, são os blocos de *Dense Connection* são usados para substituir os blocos residuais localizados na rede de *backbone PVInet*.

Cada camada do bloco de *Dense Connection* produz mapas de recursos m, que representam a taxa de crescimento. A i-ésima camada do bloco é representada por $m0 + m(i-1)$. Ele é concatenado como uma entrada. O número dos mapas de recursos de entrada da primeira camada é representado $m0$.

O bloco Connection Block (CB) apresenta cinco unidades densamente conectadas, conforme mostrado na 4.8. Cada unidade possui uma camada convolucional 1 x 1 representada pela cor cinza com o rótulo CB, que significa Normalização de Convolução em Lote com função de ativação, na 4.8 (b). Cada unidade também possui uma camada convolucional 3 x 3 represen-

tada pela cor azul na 4.8 (b), em que cada camada convolucional é seguida por uma camada de normalização em lote e a função de ativação *SRS*. O bloco amarelo denominado cc na 4.8 (b) representa a concatenação de recursos.

Nesta rede, a taxa de crescimento de *m* é definida como 32. Blocos de transição aprimorados são usados antes de cada bloco de *Dense Connection*, para realizar a etapa máxima de agrupamento e convolução. No final, ele concatena ambas as saídas como sendo a entrada do próximo bloco. Assim, os parâmetros gerais da nova rede são reduzidos.

4.2 Métricas de Validação

Métodos de classificação são aplicados a muitas aplicações em vários campos das ciências (THARWAT, 2018). Para entender o funcionamento destes métodos é preciso primeiro entender matriz de confusão, e através dela é feita a coleta de dados para serem realizados cálculos para a *accuracy* (acurácia), *recall*, *precision* (precisão), *F1-Score* e *curve roc (curva Receiver Operating Characteristic)* (FAWCETT, 2006) (THARW, 2018). A Matriz de confusão é uma matriz com a frequência de dados reais x frequência de dados de previsão pelo seu classificador. Ela é composta por Verdadeiro Positivo (*True Positive (TP)*), Falso Positivo (*False Positive (FP)*), Falso Negativo (*False Negative (FN)*) e Verdadeiro Negativo (*True Negative (TN)*).

- *TP*: Ocorre quando a frequência de dados real e a frequência de dados que estamos buscando prever foi prevista corretamente.
- *FP*: Ocorre quando a frequência de dados real e a frequência de dados que estamos buscando prever foi prevista incorretamente.
- *FN*: Ocorre quando a frequência de dados real e a frequência de dados que não estamos buscando prever foi prevista incorretamente.
- *TN*: Ocorre quando a frequência de dados real e a frequência de dados que não estamos buscando prever foi prevista corretamente.

A Tabela 4.2, ilustra como é uma tabela de confusão.

Table 4.2 – Matriz confusão

	Valores preditos	
Valores reais	TP	FP
	FN	TN

Fonte: Autor (2020)

- *Accuracy*: A *accuracy* é a métrica que calcula a taxa de acerto, o resultado consiste na divisão de todos os acertos por todos os resultados. O cálculo da *accuracy* é feito com a frequência de dados TP + TN / TP + TN + FN + TN, como é demonstrado na fórmula 4.2.

$$Accuracy = \frac{TP+TN}{TP+TN+FN+TN} \tag{4.2}$$

- *Recall*: A métrica Recall é utilizada para indicar a relação entre as previsões positivas realizadas corretamente e todas as previsões que realmente são positivas (True Positives e False Negatives). O cálculo do *recall* é feito com a frequência de dados TP/ TP + FN, como é demonstrado na fórmula 4.3.

$$Recall = \frac{TP}{TP+FN} \tag{4.3}$$

- *Precision* : A *Precision* é utilizada para indicar a relação entre as previsões positivas realizadas corretamente e todas as previsões falsas positivas. O cálculo da *Precision* é feito com a frequência de dados TP/ TP + FP, como é demonstrado na fórmula 4.4.

$$Precision = \frac{TP}{TP+FP} \tag{4.4}$$

- *F1-Score*: O *F1-Score* é a média harmônica entre o *Recall* e a *Precision*. O cálculo do *F1-Score* é feito com a frequência de dados 2 * *Precision* * *Recall* / *Precision* + *Recall*, como é demonstrado na fórmula 4.5.

$$F1-Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.5}$$

- *Curve roc*: Para realizar o cálculo da *Curve roc* é necessário ter dois dados: a *Sensitivity* (sensibilidade) e a *Specificity* (especificidade). Para obter a *sensitivity* é necessário calcular o *recall* da equação 4.3, já a *specificity* precisa do TN / FP + TN, como é demostrado na fórmula 4.6.

$$Sensitivity = recall$$
$$Specificity = \frac{TN}{FP + TN} \tag{4.6}$$

Estas métricas são utilizadas a fim de validar modelos de algoritmos para *DL*. As métricas que foram utilizadas na nossa solução foi a *Accuracy*, *Sensibilidade* e *F1-Score*, pois na literatura essas métricas são as mais utilizadas em propostas de algoritmos de *DL*

4.3 Código de Trânsito Brasileiro

O atual CTB entrou em vigor em 1998 Cidades (2008), apesar do CTB ser antigo ele, teve um grande impacto na sociedade logo após a implantação (BASTOS SELMA MAFFEI DE ANDRADE, 1999). Neste estudo eles analisaram uma redução de 18,5% o número de acidentes com vítimas. O atual CTB define que o pedestre sempre terá prioridade no trânsito, desde que ele esteja atravessando na faixa de pedestres, veículos de saúde e segurança pública também irá ter prioridades desde que estejam com suas devidas identificações, com avisos de áudio e visuais, como o acionamento da sirena característica de cada veículo e a sua adesivação.

De acordo com art. 29 do Código de Trânsito Brasileiro - Lei 9503/97, parágrafo VII (CIDADES, 2008). Os veículos destinados a bombeiros e salvamentos, veículos de polícia, vigilância e operações de trânsito e ambulâncias, além da prioridade de trânsito, desfrutam de livre circulação, estacionamento e parada em serviço de emergência e devidamente identificados pelas normas dos dispositivos para alarme sonoro e luz vermelha piscante, sujeitos às seguintes disposições:

- Quando os dispositivos são ativados, indicando a proximidade dos veículos, todos os motoristas devem deixar a passagem na faixa esquerda livre, indo para a direita da estrada e parando, se necessário;

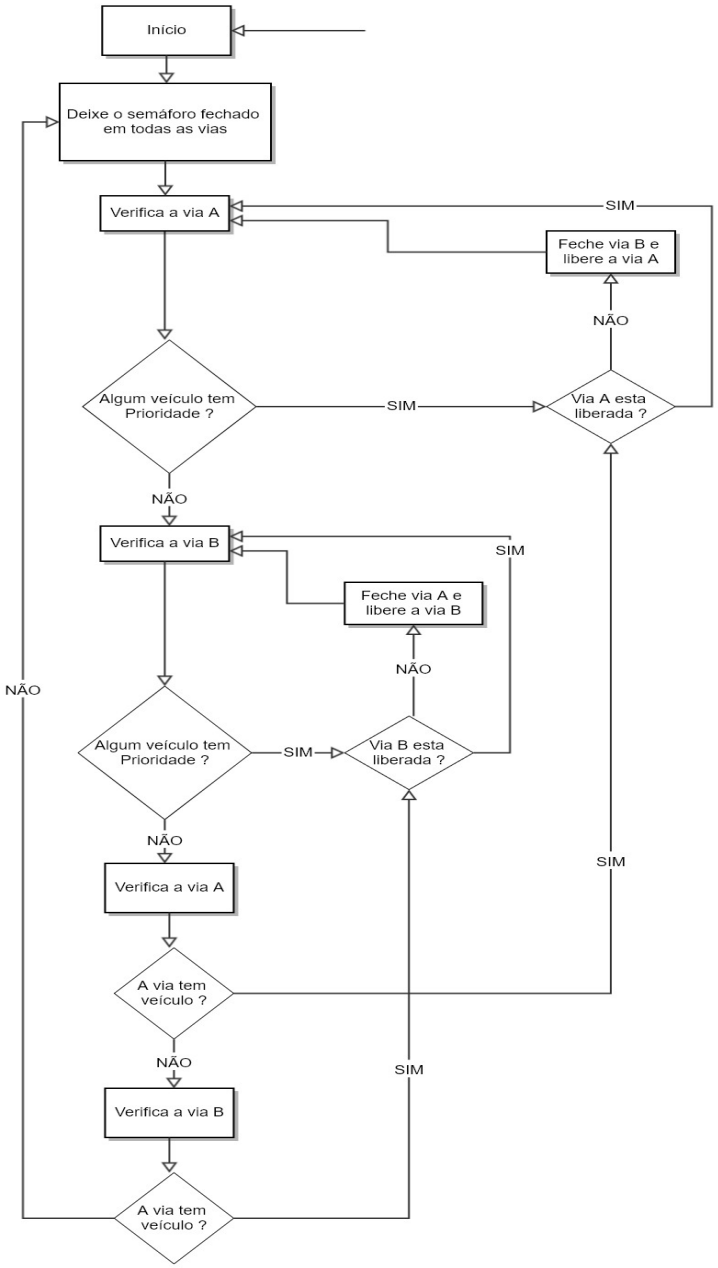
- Os pedestres, ao ouvirem o alarme sonoro, devem esperar na calçada, atravessando a rua somente quando o veículo já passou pelo local;

- O uso de dispositivos de alarme sonoro e a luz vermelha piscando só podem ocorrer quando o serviço de emergência é realmente fornecido;
- A prioridade de passagem na estrada e no cruzamento deve ser a velocidade reduzida e com as devidas precauções de segurança, em conformidade com as demais regras deste Código;

4.4 Algoritmo proposto para um semáforo inteligente

O algoritmo proposto para um semáforo inteligente irá funcionar de acordo com a Figura 4.9. Ele foi elaborado considerando o CTB Cidades (2008) e pensado no tráfego urbano brasileiro, dando prioridade no trânsito para os veículos de segurança, resgate ou de saúde que estiverem com o giroflex ligado.

Figure 4.9 – Proposta para um semáforo inteligente



Fonte: Autor (2020)

A proposta do algoritmo do semáforo inteligente funcionara na seguinte maneira:

(a) O algoritmo do semáforo irá deixar todas as vias do cruzamento fechado.

(b) O algoritmo do semáforo irá verificar se tem algum veículo de prioridade na **via A**, se o algoritmo não conseguir identificar nenhum veículo de prioridade o próximo passo será o processo 3, caso ele consiga identificar prioridade o próximo passo será o processo 6.

(c) O algoritmo do semáforo irá verificar se tem algum veículo de prioridade na **via B**, se o algoritmo não conseguir identificar nenhum veículo de prioridade o próximo passo será o processo 2, caso ele consiga identificar prioridade o próximo passo será o processo 7.

(d) O algoritmo do semáforo irá verificar se tem algum veículo na **via A**, se o algoritmo não conseguir identificar nenhum veículo próximo passo será o processo 5.

(e) O algoritmo do semáforo irá verificar se tem algum veículo na **via B**, se o algoritmo não conseguir identificar nenhum veículo próximo passo será o processo 1.

(f) O algoritmo do semáforo irá fechar o semáforo da **via B** e liberar a **via A**, após este processo, próximo passo será o processo 2.

(g) O algoritmo do semáforo irá fechar o semáforo da **via A** e liberar a **via B**, após este processo, próximo passo será o processo 3.

Na solução proposta o semáforo irá ter suas particularidades, onde ele dará prioridade para alguns veículos especiais que trafeguem na via. O semáforo poderá gerenciar isso de maneira automática. É importante destacar que este algoritmo proposto não foi implementado no presente trabalho, mas pode servir como base de trabalhos futuros.

A prioridade dos veículos será da seguinte maneira:

Table 4.3 – Prioridade do Semáforo

Prioridade do Semáforo	Veículos
1	Veículos de Socorro em serviço
2	Veículos de Segurança Pública em serviço
3	Veículos de Transporte Urbano
4	Veículos Populares

Fonte: Adaptado do CTB (2020)

Se o cruzamento tiver dois veículos de prioridade, o semáforo irá abrir a via para o veículo de prioridade maior ou irá liberar a passagem para o veículo que o DL conseguir identificar primeiro. Caso ambos os veículos tenham a mesma prioridade, então o semáforo irá seguir o CTB dando a prioridade para o lado esquerdo do cruzamento.

4.5 Ferramentas

A ferramenta utilizada foi o $Matlab2019_a$, que é uma poderosa ferramenta matemática de desenvolvimento. Ela contém um ambiente de programação de nível alto, o qual facilita a interação e utilização do usuário e uma flexibilidade de programação.

Ao contrário de linguagens de programação tradicionais como o C ou o *Delphi*, no *Matlab* o usuário não precisa ficar preocupado com alocação de memória ou com a declaração de variáveis, além de apresentar uma série de funções matemáticas já implementadas.

A solução proposta foi implementada utilizando estrutura *TensorFlow* que é uma biblioteca de software com o código aberto para computação numérica usando grafos computacionais. Foi originalmente desenvolvido pela organização de pesquisa de Inteligência de máquina do Google o *Google Brain Team* esta organização consiste em pesquisas de *DL*, juntamente com o *Keras* que é uma API de *DL* escrita em Python, que é executada na plataforma de máquina *TensorFlow*.

Os modelos de detecção foram treinados e testados usando um servidor *NVIDIA Titan X*.

5 RESULTADOS

Tendo em vista que o *YOLOv3* vem sendo muito utilizado na literatura em aplicações em tempo real, obtendo valores na acurácia e o tempo de execução altamente favoráveis, resolvemos utilizar este algoritmo e o *PVInet* que é uma proposta de arquitetura baseada no *YOLOv3* junto com as funções de ativação mais utilizadas em DL que são a *Softmax*, *ReLU* e a *SRS*.

As Tabelas 5.1, 5.2, 5.3, 5.4 e 5.5 apresentam os resultados da avaliação de desempenho do *PVInet* proposto com diferentes funções de ativação em relação ao *YOLOv3* para classificar as imagens na fase de treinamento de cada classe de veículo.

Para o treinamento da solução foram utilizadas 3720 imagens, o que equivale a 80% do BD e para testes foram utilizadas 930 imagens o que equivale a 20% do BD.

Em nossa solução, foi utilizada, uma validação cruzada k-fold, para k = 10. Assim, a validação cruzada de 10 vezes foi empregada para obter as métricas para validação da classificação do veículo.

A Tabela 5.1 demostra os Autor que foram gerados utilizando imagens da BD direita, esquerda e frente respectivamente, na categoria de Ambulância. As métricas de validação utilizadas nesta fase de treinamento foram Acurácia, Sensibilidade e *F1-Score*.

Table 5.1 – Resultados do treinamento da classe **Ambulância**

Modelo	Posição da imagem	Acurácia	Sensibilidade	F1-Score
YOLOv3	Di / Es / Fr	0,85/0,84/0,85	0,85/0,85/0,84	0,85/0,84/0,84
PVInet (Softmax)	Di / Es / Fr	0,87/0,87/0,88	0,88/0,88/0,87	0,87/0,87/0,87
PVInet (ReLU)	Di / Es / Fr	0,91/0,90/0,92	0,89/0,90/0,90	0,89/0,90/0,90
PVInet (SRS)	**Di / Es / Fr**	**0,93/0,94/0,94**	**0,93/0,93/0,93**	**0,93/0,93/0,93**

Fonte: Autor (2020)

Utilizando o PVInet com a função de ativação SRS, a acurácia na fase de treinamentos da classe de **Ambulância** obteve resultados de 0,94% com as imagens posicionadas a Es e Fr e 0,93% para imagens posicionadas a Di. As métricas de validação Sensibilidade e *F1-Score* obtiveram os resultados de 0,93% em todas as posições das imagens.

A Tabela 5.2 demostra os Autor que foram gerados utilizando imagens da BD direita, esquerda e frente respectivamente, na categoria de Caminhão de Bombeiros. As métricas de validação utilizadas nesta fase de treinamento foram Acurácia, Sensibilidade e *F1-Score*.

Table 5.2 – Resultados do treinamento da classe **Caminhão de Bombeiros**

Modelo	Posição da imagem	Acurácia	Sensibilidade	F1-Score
YOLOv3	Di / Es / Fr	0,88/0,87/0,88	0,87/0,88/0,87	0,87/0,87/0,87
PVInet (Softmax)	Di / Es / Fr	0,91/0,90/0,90	0,89/0,90/0,89	0,89/0,89/0,89
PVInet (ReLU)	Di / Es / Fr	0,93/0,91/0,92	0,92/0,91/0,92	0,92/0,91/0,92
PVInet (SRS)	**Di / Es / Fr**	**0,95/0,95/0,94**	**0,94/0,94/0,95**	**0,94/0,94/0,94**

Fonte: Autor (2020)

Utilizando o PVInet com a função de ativação SRS, a acurácia na fase de treinamentos da classe de **Caminhão de Bombeiros** obteve resultados de 0,95% com as imagens posicionadas a Di e Es e 0,94% para imagens posicionadas a Fr. A métrica de validação de Sensibilidade obtive os resultados de 0,95% com as imagens posicionadas a Fr e 0,94% com as imagens posicionadas a Di e Es, a métrica de validação *F1-Score* obteve os resultados de 0,94% em todas as posições das imagens.

A Tabela 5.3 demostra os Autor que foram gerados utilizando imagens da BD direita, esquerda e frente respectivamente, na categoria de Carro. As métricas de validação utilizadas nesta fase de treinamento foram Acurácia, Sensibilidade e *F1-Score*.

Table 5.3 – Resultados do treinamento da classe **Carro**

Modelo	Posição da imagem	Acurácia	Sensibilidade	F1-Score
YOLOv3	Di / Es / Fr	0,82/0,81/0,81	0,81/0,81/0,82	0,81/0,81/0,81
PVInet (Softmax)	Di / Es / Fr	0,85/0,84/0,85	0,85/0,85/0,86	0,85/0,84/0,85
PVInet (ReLU)	Di / Es / Fr	0,89/0,88/0,89	0,89/0,88/0,89	0,89/0,88/0,89
PVInet (SRS)	**Di / Es / Fr**	**0,92/0,92/0,92**	**0,92/0,92/0,93**	**0,92/0,92/0,92**

Fonte: Autor (2020)

Utilizando o PVInet com a função de ativação SRS, a acurácia na fase de treinamentos da classe de **Carro** obteve resultados de 0,92% em todas as posições das imagens. A métrica de validação Sensibilidade obteve os resultados de 0,93% com as imagens posicionadas a Fr e 0,92% com as imagens posicionadas a Di e Es, a métrica de validação de *F1-Score* obteve os resultados de 0,92% em todas as posições das imagens.

A Tabela 5.4 demostra os Autor que foram gerados utilizando imagens da BD direita, esquerda e frente respectivamente, na categoria de Ônibus. As métricas de validação utilizadas nesta fase de treinamento foram Acurácia, Sensibilidade e *F1-Score*.

Table 5.4 – Resultados do treinamento da classe **Ônibus**

Modelo	Posição da imagem	Acurácia	Sensibilidade	F1-Score
YOLOv3	Di / Es / Fr	0,86/0,87/0,87	0,85/0,86/0,86	0,85/0,86/0,86
PVInet (Softmax)	Di / Es / Fr	0,89/0,89/0,90	0,88/0,90/0,88	0,88/0,89/0,88
PVInet (ReLU)	Di / Es / Fr	0,91/0,91/0,92	0,91/0,91/0,92	0,91/0,91/0,92
PVInet (SRS)	**Di / Es / Fr**	**0,94/0,93/0,94**	**0,94/0,93/0,95**	**0,94/0,93/0,94**

Fonte: Autor (2020)

Utilizando o PVInet com a função de ativação SRS, a acurácia na fase de treinamentos da classe de **Ônibus** obteve resultados de 0,94% com as imagens posicionadas a Di e Fr e 0,93% com as imagens posicionadas a Es. A métrica de validação Sensibilidade obteve resultado de 0,95% com as imagens posicionadas a Fr, 0,94% com as imagens posicionadas a Di e 0,93% com as imagens posicionadas a Es, a métrica de validação de *F1-Score* obteve os resultados de 0,94% com as imagens posicionadas a Di e Fr, 0,93% com as imagens posicionadas a Es.

A Tabela 5.5 demostra os Autor que foram gerados utilizando imagens da BD direita, esquerda e frente respectivamente, na categoria de Ambulância. As métricas de validação utilizadas nesta fase de treinamento foram Acurácia, Sensibilidade e *F1-Score*.

Table 5.5 – Resultados do treinamento da classe **Polícia**

Modelo	Posição da imagem	Acurácia	Sensibilidade	F1-Score
YOLOv3	Di / Es / Fr	0,83/0,84/0,83	0,82/0,83/0,82	0,83/0,84/0,83
PVInet (Softmax)	Di / Es / Fr	0,86/0,86/0,87	0,86/0,87/0,86	0,86/0,87/0,87
PVInet (ReLU)	Di / Es / Fr	0,90/0,89/0,91	0,89/0,90/0,90	0,90/0,90/0,90
PVInet (SRS)	**Di / Es / Fr**	**0,92/0,93/0,93**	**0,92/0,93/0,93**	**0,92/0,93/0,93**

Fonte: Autor (2020)

Utilizando o PVInet com a função de ativação SRS, a acurácia na fase de treinamentos da classe de **Polícia** obteve resultados de 0,93% com as imagens posicionadas a Es e Fr e 0,92% com as imagens posicionadas a Di. A métrica de validação Sensibilidade obteve resultado de 0,93% com as imagens posicionadas a Es e Fr, 0,92% com as imagens posicionadas a Di, a métrica de validação de *F1-Score* obteve os resultados de 0,93% com as imagens posicionadas a Es e Fr, 0,92% com as imagens posicionadas a Di.

A Tabela 5.6 apresenta os resultados da avaliação de desempenho de nossa proposta na fase de treinamento, utilizando imagens de caminhão de bombeiro, ônibus, ambulância, polícia e carro do conjunto da BD onde estas imagens são separadas em Di , Es e Fr.

Nesta fase de testes de precisão juntamos, todas as subcategorias das imagens de cada classe e denominamos de Subcategoria Junta (SJ), que consiste em utilizar todas as imagens de cada classe, independente da posição da imagem. A métrica de validação utilizada nesta fase de treinamento foi a Acurácia.

Table 5.6 – Resultados da acurácia com a BD na fase de treinamento utilizando o SJ

Modelo	Posição da imagem	Caminhão de Bombeiros	Ônibus	Ambulância	Polícia	Carro
YOLOv3	SJ	0.90	0.88	0.87	0.85	0.84
PVInet (Softmax)	SJ	0.92	0.91	0.89	0.87	0.86
PVInet (ReLU)	SJ	0.93	0.92	0.91	0.89	0.88
PVInet (SRS)	**SJ**	**0.98**	**0.98**	**0.96**	**0.95**	**0.95**

Fonte: Autor (2020)

Utilizando o PVInet com a função de ativação SRS, a acurácia na fase de treinamentos com todas as posições das imagens, a classe de **Caminhão de Bombeiros** e **Ônibus** obteve o resultado de 0,98%, 0,96% na classe de **Ambulância** e 0,95% nas classes de **Polícia** e **Carro**.

As Tabelas 5.7, 5.8, 5.9, 5.10 e 5.11 apresentam os resultados da avaliação de desempenho do *PVInet* proposto com diferentes funções de ativação em relação ao *YOLOv3* para classificar as imagens na fase de testes para cada classe de veículo.

A Tabela 5.7 demostra os resultados dos testes que foram gerados utilizando imagens da BD direita, esquerda e frente respectivamente, na categoria de Ambulância. As métricas de validação utilizadas nesta fase de testes foram Acurácia, Sensibilidade e *F1-Score*.

Table 5.7 – Resultados dos testes da classe **Ambulância**

Modelo	Posição da imagem	Acurácia	Sensibilidade	F1-Score
YOLOv3	Di / Es / Fr	0.84/0.84/0.85	0.84/0.85/0.84	0.84/0.84/0.84
PVInet (Softmax)	Di / Es / Fr	0.87/0.87/0.88	0.87/0.88/0.87	0.87/0.87/0.87
PVInet (ReLU)	Di / Es / Fr	0.91/0.90/0.92	0.89/0.90/0.90	0.89/0.90/0.90
PVInet (SRS)	**Di / Es / Fr**	**0.93/0.94/0.93**	**0.93/0.93/0.93**	**0.93/0.93/0.93**

Fonte: Autor (2020)

Utilizando o PVInet com a função de ativação SRS, a acurácia na fase de testes da classe de **Ambulância** obteve resultados de 0,94% com as imagens posicionadas a Es, 0,93% com as imagens posicionadas a Di e Fr. As métricas de validação de Sensibilidade e *F1-Score* obtiveram os resultados de 0,93% em todas as posições das imagens.

A Tabela 5.8 demostra os resultados dos testes que foram gerados utilizando imagens da BD direita, esquerda e frente respectivamente, na categoria de Caminhão de Bombeiro. As métricas de validação utilizadas nesta fase de testes foram Acurácia, Sensibilidade e *F1-Score*.

Table 5.8 – Resultados dos testes da classe **Caminhão de Bombeiros**

Modelo	Posição da imagem	Acurácia	Sensibilidade	F1-Score
YOLOv3	Di / Es / Fr	0.87/0.87/0.88	0.87/0.86/0.87	0.87/0.87/0.87
PVInet (Softmax)	Di / Es / Fr	0.90/0.90/0.90	0.89/0.90/0.89	0.90/0.89/0.89
PVInet (ReLU)	Di / Es / Fr	0.93/0.92/0.92	0.92/0.91/0.92	0.92/0.92/0.92
PVInet (SRS)	**Di / Es / Fr**	**0.95/0.94/0.94**	**0.94/0.94/0.95**	**0.94/0.94/0.94**

Fonte: Autor (2020)

Utilizando o PVInet com a função de ativação SRS, a acurácia na fase de testes da classe de **Caminhão de Bombeiros** obteve resultados de 0,95% com as imagens posicionadas a Di, 0,94% com as imagens posicionadas a Es e Fr. As métricas de validação de Sensibilidade obteve resultados de 0,95% com as imagens posicionadas a Di, 0,94% com as imagens posicionadas a Es e Fr e a métrica de validação *F1-Score* obtiveram os resultados de 0,94% em todas as posições das imagens.

A Tabela 5.9 demostra os resultados dos testes que foram gerados utilizando imagens da BD direita, esquerda e frente respectivamente, na categoria de Carro. As métricas de validação utilizadas nesta fase de testes foram Acurácia, Sensibilidade e *F1-Score*.

Table 5.9 – Resultados dos testes da classe **Carro**

Modelo	Posição da imagem	Acurácia	Sensibilidade	F1-Score
YOLOv3	Di / Es / Fr	0.79/0.81/0.81	0.80/0.81/0.82	0.79/0.81/0.81
PVInet (Softmax)	Di / Es / Fr	0.84/0.84/0.85	0.85/0.85/0.86	0.84/0.84/0.85
PVInet (ReLU)	Di / Es / Fr	0.88/0.88/0.89	0.88/0.88/0.89	0.88/0.88/0.89
PVInet (SRS)	**Di / Es / Fr**	**0.92/0.91/0.92**	**0.92/0.92/0.93**	**0.92/0.91/0.92**

Fonte: Autor (2020)

Utilizando o PVInet com a função de ativação SRS, a acurácia na fase de testes da classe de **Carro** obteve resultados de 0,92% com as imagens posicionadas a Di e Fr, 0,91% com as imagens posicionadas a Es. As métricas de validação de Sensibilidade obtiveram resultados de 0,93% com as imagens posicionadas a Fr, 0,92% com as imagens posicionadas a Di e Es, a métrica de validação *F1-Score* obtiveram os resultados de 0,92% com as imagens posicionadas a Di e Fr, 0,91% com as imagens posicionadas a Es.

A Tabela 5.10 demostra os resultados dos testes que foram gerados utilizando imagens da BD direita, esquerda e frente respectivamente, na categoria de Ônibus. As métricas de validação utilizadas nesta fase de testes foram Acurácia, Sensibilidade e *F1-Score*.

Table 5.10 – Resultados dos testes da classe **Ônibus**

Modelo	Posição da imagem	Acurácia	Sensibilidade	F1-Score
YOLOv3	Di / Es / Fr	0.86/0.86/0.87	0.85/0.86/0.86	0.85/0.86/0.86
PVInet (Softmax)	Di / Es / Fr	0.90/0.89/0.90	0.88/0.90/0.88	0.89/0.89/0.88
PVInet (ReLU)	Di / Es / Fr	0.90/0.91/0.92	0.91/0.91/0.92	0.90/0.91/0.92
PVInet (SRS)	**Di / Es / Fr**	**0.93/0.93/0.95**	**0.94/0.93/0.95**	**0.93/0.93/0.95**

Fonte: Autor (2020)

Utilizando o PVInet com a função de ativação SRS, a acurácia na fase de testes da classe de **Ônibus** obteve resultados de 0,95% com as imagens posicionadas a Fr, 0,93% com as imagens posicionadas a Di e Es. As métricas de validação de Sensibilidade obteve resultados de 0,95% com as imagens posicionadas a Fr, 0,94% com as imagens posicionadas a Di e 0,93% com as imagens posicionadas Es, a métrica de validação *F1-Score* obtiveram os resultados de 0,95% com as imagens posicionadas a Fr, 0,93% com as imagens posicionadas a Di e Es.

A Tabela 5.11 demostra os resultados dos testes que foram gerados utilizando imagens da BD direita, esquerda e frente respectivamente, na categoria de Polícia. As métricas de validação utilizadas nesta fase de testes foram Acurácia, Sensibilidade e *F1-Score*.

Table 5.11 – Resultados dos testes da classe **Polícia**

Modelo	Posição da imagem	Acurácia	Sensibilidade	F1-Score
YOLOv3	Di / Es / Fr	0.82/0.84/0.83	0.82/0.83/0.82	0.82/0.84/0.83
PVInet (Softmax)	Di / Es / Fr	0.86/0.86/0.87	0.86/0.87/0.86	0.86/0.87/0.87
PVInet (ReLU)	Di / Es / Fr	0.90/0.89/0.91	0.90/0.90/0.90	0.90/0.90/0.90
PVInet (SRS)	**Di / Es / Fr**	**0.92/0.93/0.93**	**0.92/0.92/0.93**	**0.92/0.92/0.93**

Fonte: Autor (2020)

Utilizando o PVInet com a função de ativação SRS, a acurácia na fase de testes da classe de **Polícia** obteve resultados de 0,93% com as imagens posicionadas a Es e Fr, 0,92% com as imagens posicionadas a Di. As métricas de validação de Sensibilidade obtiveram resultados de 0,93% com as imagens posicionadas a Fr, 0,92% com as imagens posicionadas a Di e Es, a métrica de validação *F1-Score* obtiveram os resultados de 0,93% com as imagens posicionadas a Fr, 0,92% com as imagens posicionadas a Di e Es.

A Tabela 5.12 apresenta os resultados da avaliação de desempenho de nossa proposta na fase de testes, utilizando imagens de caminhão de bombeiro, ônibus, ambulância, polícia e carro do conjunto da BD onde é composto por imagens direita, esquerda e frontal. A métrica de validação utilizada nesta fase de teste foi a Acurácia.

Table 5.12 – Resultados da acurácia com a BD na fase de testes utilizando o SJ

Modelo	Posição da imagem	Caminhão de Bombeiros	Ônibus	Ambulância	Polícia	Carro
YOLOv3	SJ	0.89	0.87	0.87	0.84	0.84
PVInet (Softmax)	SJ	0.92	0.92	0.89	0.86	0.86
PVInet (ReLU)	SJ	0.94	0.92	0.91	0.88	0.87
PVInet (SRS)	**SJ**	**0.97**	**0.97**	**0.96**	**0.94**	**0.93**

Fonte: Autor (2020)

Utilizando o PVInet com a função de ativação SRS, a acurácia na fase de testes com todas as posições das imagens de cada classe. A classe de **Caminhão de Bombeiros** e **Ônibus** obteve o resultado de 0,97%, 0,96% na classe de **Ambulância** e 0,94% na classe de **Polícia** e 0,93% na classe de **Carro**.

Após a identificação destes veículos utilizando o *PVInet* com a função de ativação SRS, o semáforo determinará se irá liberar a movimentação de veículos no cruzamento ou não. A solução proposta é que a identificação destes veículos seja em tempo real, pois será utilizada junto com um semáforo inteligente. Na seção 4.4, foi proposto um algoritmo para um semáforo inteligente.

5.1 Limitações

O escopo de nossa proposta de semáforo inteligente terá as seguintes limitações:

- Segundo o CTB, veículos de polícia e/ou ambulância não estando a serviço não possuem prioridade maior que pedestre. No presente trabalho, sensores sonoros não são utilizados.
- Os modelos de veículos de saúde (ambulância) e/ou segurança pública (policiamento) variam para cada país, e inclusive para cada cidade, Neste trabalho, assumiremos as versões que tem nas BD.
- O algoritmo de prioridade do semáforo foi desenvolvido considerando somente um cruzamento de duas vias.

6 CONCLUSÃO

Nesta dissertação, foi proposto um modelo de detecção de veículos, que foi estudado e implementado. Métodos como extração de características e ativação de funções foram investigados.

Foi realizada uma longa pesquisa sobre algoritmos de *DL*, na qual constatamos que o algoritmo $YOLO_{V3}$ original tem alto desempenho em aplicações em tempo real e vem sendo muito utilizado em diversas aplicações com o *DL*. Por este motivo, resolvemos escolher o $YOLO_{V3}$ para melhorar o procedimento de detecção e o tempo. Foi proposta uma versão aprimorada do $YOLO_{V3}$, o *PVInet*, que reduz o número de parâmetros em comparação com o $YOLO_{V3}$, deixando o *PVInet* muito mais eficiente, melhorando seu desempenho e sua velocidade. Também utilizamos as funções de ativação *Softmax*, *ReLU* e *SRS*, onde a função *SRS* apresentou uma acurácia de 0,98% na fase de treinamento e uma acurácia de até 0,97% na fase de testes.

Foi proposta também a criação de um algoritmo de semáforo inteligente, ao qual *PVInet* irá detectar os veículos no trânsito e o semáforo inteligente irá determinar e gerenciar o que deverá ser feito em tempo real.

Para melhorar o procedimento de detecção e a execução do tempo, foi proposta uma versão aprimorada do $YOLO_{V3}$, o PVInet no qual consiste em uma concatenação no número de parâmetros do $YOLO_{V3}$ o qual reduz o número de parâmetros fazendo que esta proposta aprimora uma propagação de recursos e o desempenho da rede. Utilizando a função de ativação *SRS* em nossa proposta, conseguimos resultados melhores e um tempo de execução melhor que o modelo $YOLO_{V3}$ original.

Os resultados experimentais demonstraram que o *PVInet* apresentou baixa complexidade e alta velocidade de detecção de semáforos inteligentes. Foi desenvolvida a criação de uma BD com imagens homogêneas de veículos brasileiros. Para trabalhos futuros, o número de imagens desta BD será ampliado e testaremos esta proposta em um ambiente real.

Sabemos que há muito para ser feito, como o aumento de detecção dos veículos no trânsito como motos, ciclistas e pedestres, melhoramento do algoritmo de semáforo inteligente e realização de testes para averiguar qual a melhor forma de posicionamento das câmeras, as quais são utilizadas na captação das imagens para que o modelo *PVInet* possa detectar veículos e o semáforo possa ser capaz de determinar o que deve ser feito.

REFERÊNCIAS

ADARSH, P.; RATHI, P.; KUMAR, M. Yolo v3-tiny: Object detection and recognition using one stage improved model. In: . [S.l.]: IEEE, 2020. p. 687–694. ISBN 978-1-7281-5198-4. ISSN 2469-5556.

ALAM, M. et al. Real-time smart parking systems integration in distributed its for smart cities. **Journal of Advanced Transportation**, Hindawi, v. 2018, 2018.

ALPAYDIN, E. **Introduction to Machine Learning**. [S.l.]: Massachusetts Institute of Technology, 2010.

ANGELIDOU, M. Smart city policies: A spatial approach. **Cities**, v. 41, p. S3 – S11, 2014. ISSN 0264-2751. Current Research on Cities. Disponível em: <http://www.sciencedirect.com/science/article/pii/S026427511400095X>.

ASH, R. et al. Real-time pedestrian traffic light detection. In: . [S.l.]: IEEE, 2018. p. 1–5. ISBN 978-1-5386-6379-0.

ASHA, C. S.; NARASIMHADHAN, A. V. Vehicle counting for traffic management system using yolo and correlation filter. In: . [S.l.]: IEEE, 2018. p. 1–6. ISBN 978-1-5386-1113-5.

BADRINARAYANAN, V.; KENDALL, A.; CIPOLLA, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 39, n. 12, p. 2481–2495, 2017.

BARBER, D. **Bayesian Reasoning and Machine Learning**. [S.l.]: Cambridge University Press, 2012.

BASTOS SELMA MAFFEI DE ANDRADE, L. C. J. Y. G. L. Acidentes de trânsito e o novo código de trânsito brasileiro em cidade da região sul do brasil. 1999.

BAUZA, R.; GOZALVEZ, J.; SANCHEZ-SORIANO, J. Road traffic congestion detection through cooperative vehicle-to-vehicle communications. In: **IEEE Local Computer Network Conference**. [S.l.: s.n.], 2010. p. 606–612.

BEHRENDT, K.; NOVAK, L.; BOTROS, R. A deep learning approach to traffic lights: Detection, tracking, and classification. In: **Proc. IEEE Int. Conf. Robotics and Automation (ICRA)**. [S.l.: s.n.], 2017. p. 1370–1377.

BENGIO, Y. **Learning Deep Architectures for AI**. Now Publishers, 2009. 131 p. (Essence of knowledge, The). ISBN 9781601982940. Disponível em: <https://books.google.com.br/books?id=cq5ewg7FniMC>.

BISHOP, C. Pattern recognition and machine learning. In: ____. [S.l.: s.n.], 2006. v. 16, p. 140–155.

CEBALLOS, G. R.; LARIOS, V. M. A model to promote citizen driven government in a smart city: Use case at gdl smart city. In: **2016 IEEE International Smart Cities Conference (ISC2)**. [S.l.: s.n.], 2016. p. 1–6.

CHAPELLE, O.; SCHÖLKOPF, B.; ZIEN, A. **Semi-supervised Learning**. MIT Press, 2010. (Adaptive computation and machine learning). ISBN 9780262514125. Disponível em: <https://books.google.com.br/books?id=zHAOQgAACAAJ>.

CHEN, F.; HE, G.; CHEN, G. Realization of boolean functions via cnn: Mathematical theory, lsbf and template design. **IEEE Transactions on Circuits and Systems I: Regular Papers**, v. 53, n. 10, p. 2203–2213, 2006.

CHEN, S.; LIN, W. Embedded system real-time vehicle detection based on improved yolo network. In: . [S.l.]: IEEE, 2019. p. 1400–1403. ISBN 978-1-7281-0514-7.

CHEN, W.; YEO, C. K. Unauthorized parking detection using deep networks at real time. In: IEEE. **2019 IEEE International Conference on Smart Computing (SMARTCOMP)**. [S.l.], 2019. p. 459–463.

CHOLLET, F. **Deep Learning with Python**. Manning Publications Company, 2018. ISBN 9781617294433. Disponível em: <https://books.google.com.br/books?id=Yo3CAQAACAAJ>.

CHOUDEKAR, P.; BANERJEE, S.; MUJU, M. K. Implementation of image processing in real time traffic light control. In: **Proc. 3rd Int. Conf. Electronics Computer Technology**. [S.l.: s.n.], 2011. v. 2, p. 94–98.

CHOURABI, H. et al. Understanding smart cities: An integrative framework. In: **Proc. 45th Hawaii Int. Conf. System Sciences**. [S.l.: s.n.], 2012. p. 2289–2297. ISSN 1530-1605.

CHUA, L. O. **CNN: A Paradigm for Complexity**. WORLD SCIENTIFIC, 1998. Disponível em: <https://www.worldscientific.com/doi/abs/10.1142/3801>.

CIABURRO, B. V. G. **Neural Networks with R**. [S.l.: s.n.], 2017.

CIDADES, C. N. d. T. e. D. N. d. T. Ministério das. **Código de trânsito brasileito e legislação complementar em vigor**. [S.l.: s.n.], 2008.

CLEVERT, D.-A.; UNTERTHINER, T.; HOCHREITER, S. Fast and accurate deep network learning by exponential linear units (elus). **Under Review of ICLR2016 (1997)**, 11 2015.

CREVIER, D. **AI: The Tumultuous History of the Search for Artificial Intelligence**. [S.l.: s.n.], 1993. 1-386 p.

DASGUPTA, M.; BANDYOPADHYAY, O.; CHATTERJI, S. Automated helmet detection for multiple motorcycle riders using cnn. In: . [S.l.]: IEEE, 2019. p. 1–4. ISBN 978-1-7281-5399-5.

DENG, L. A tutorial survey of architectures, algorithms, and applications for deep learning. **APSIPA Transactions on Signal and Information Processing**, Cambridge University Press, v. 3, p. e2, 2014.

DU, L. et al. Real-time detection of vehicle and traffic light for intelligent and connected vehicles based on yolov3 network. In: . [S.l.]: IEEE, 2019. p. 388–392. ISBN 978-1-7281-0490-4.

DÍAZ, N.; GUERRA, J.; NICOLA, J. Smart traffic light control system. In: **2018 IEEE Third Ecuador Technical Chapters Meeting (ETCM)**. [S.l.: s.n.], 2018. p. 1–4.

FAWCETT, T. An introduction to roc analysis. **ELSEVIER**, 2006.

GIFFINGER RUDOLF; FERTNER, C. K. H. K. R. P.-M. N. M. E. Smart cities - ranking of european medium-sized cities. **UNIVERSITY OF COPENHAGEN**, 2007.

GIRSHICK, R. Fast r-cnn. **Microsoft Research**, 2015.

GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2014. p. 580–587.

GONG, H. et al. Object detection based on improved yolov3-tiny. In: . [S.l.]: IEEE, 2019. p. 3240–3245. ISBN 978-1-7281-4095-7. ISSN 2688-092X.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GOODFELLOW, I. et al. Maxout networks. **30th International Conference on Machine Learning, ICML 2013**, v. 1302, 02 2013.

GU, S. et al. A new deep learning method based on alexnet model and ssd model for tennis ball recognition. In: **2017 IEEE 10th International Workshop on Computational Intelligence and Applications (IWCIA)**. [S.l.: s.n.], 2017. p. 159–164.

HARAHAP, E. et al. Modeling and simulation of queue waiting time at traffic light intersection. **Journal of Physics: Conference Series**, IOP Publishing, v. 1188, p. 012001, mar 2019. Disponível em: <https://doi.org/10.1088%2F1742-6596%2F1188%2F1%2F012001>.

HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: **2015 IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2015. p. 1026–1034.

HE, K. et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

HINTON, G. E. Deep belief networks. **Scholarpedia**, v. 4, p. 5947, 2009.

HOLANDA, A. B. de. **Novo Dicionário Aurélio Da Língua Portuguesa (1986)**. [S.l.]: 1 janeiro 1986, 1986. ISBN 8520904114.

HUANG, G. et al. Densely connected convolutional networks. In: **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. p. 2261–2269.

HUANG, J. et al. Speed/accuracy trade-offs for modern convolutional object detectors. In: . [S.l.]: IEEE, 2017. p. 3296–3297. ISBN 978-1-5386-0458-8. ISSN 1063-6919.

HUANG, X. et al. The apolloscape dataset for autonomous driving. In: . [S.l.]: IEEE, 2018. p. 1067–10676. ISBN 978-1-5386-6101-7. ISSN 2160-7508.

HUANG, Y.-Q. et al. Optimized yolov3 algorithm and its application in traffic flow detections. **Applied Sciences**, v. 10, p. 3079, 04 2020.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. **CoRR**, abs/1502.03167, 2015. Disponível em: <http://arxiv.org/abs/1502.03167>.

JEONG, J.; PARK, H.; KWAK, N. Enhancement of ssd by concatenating feature maps for object detection. **arXiv preprint arXiv:1705.09587**, 2017.

JIAO, L. et al. A survey of deep learning-based object detection. IEEE, v. 7, p. 128837–128868, 2019. ISSN 2169-3536.

KANUNGO, A.; SHARMA, A.; SINGLA, C. Smart traffic lights switching and traffic density calculation using video processing. In: . [S.l.]: IEEE, 2014. p. 1–6. ISBN 978-1-4799-2290-1.

KLAMBAUER, G. et al. Self-normalizing neural networks. **CoRR**, abs/1706.02515, 2017. Disponível em: <http://arxiv.org/abs/1706.02515>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. Imagenet classification with deep convolutional neural networks. **Neural Information Processing Systems**, v. 25, 01 2012.

KUMAR, N. M.; GOEL, S.; MALLICK, P. K. Smart cities in india: Features, policies, current status, and challenges. In: **2018 Technologies for Smart-City Energy Security and Power (ICSESP)**. [S.l.: s.n.], 2018. p. 1–4.

LAM, C.-T.; NG, B.; CHAN, C.-W. Real-time traffic status detection from on-line images using generic object detection system with deep learning. In: . [S.l.]: IEEE, 2019. p. 1506–1510. ISBN 978-1-7281-0536-9. ISSN 2576-7844.

LAN, W. et al. Pedestrian detection based on yolo network model. In: . [S.l.]: IEEE, 2018. p. 1547–1551. ISBN 978-1-5386-6076-8. ISSN 2152-7431.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, p. 436–44, 05 2015.

LEE, J. T.; CHUNG, Y. Deep learning-based vehicle classification using an ensemble of local expert and global networks. In: **Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops (CVPRW)**. [S.l.: s.n.], 2017. p. 920–925. ISSN 2160-7516.

LI, J. et al. Application research of improved yolo v3 algorithm in pcb electronic component detection. **Applied Sciences**, Multidisciplinary Digital Publishing Institute, v. 9, n. 18, p. 3750, 2019.

LIN, J.-P.; SUN, M.-T. A yolo-based traffic counting system. In: . [S.l.]: IEEE, 2018. p. 82–85. ISBN 978-1-7281-1230-5. ISSN 2376-6816.

LIU, W. et al. Ssd: Single shot multibox detector. In: LEIBE, B. et al. (Ed.). **Computer Vision – ECCV 2016**. Cham: Springer International Publishing, 2016. p. 21–37. ISBN 978-3-319-46448-0.

LIU, Y.; LIU, L.; CHEN, W. Intelligent traffic light control using distributed multi-agent q learning. In: **Proc. IEEE 20th Int. Conf. Intelligent Transportation Systems (ITSC)**. [S.l.: s.n.], 2017. p. 1–8. ISSN 2153-0017.

LOU, L. et al. Vehicles detection of traffic flow video using deep learning. In: . [S.l.]: IEEE, 2019. p. 1012–1017. ISBN 978-1-7281-1455-2.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: **in ICML Workshop on Deep Learning for Audio, Speech and Language Processing**. [S.l.: s.n.], 2013.

MANVILLE, C. et al. **Mapping Smart Cities in the EU. Study**. [S.l.]: Directorate-General for Internal Policies, Policy Department A: Economic and . . . , 2014.

MCCARTHY JOHN; MINSKY, M. L. R. N.; SHANNON, C. E. A proposal for the dartmouth summer research project on artificial intelligence. **AI Magazine**, v. 27, n. 4, 2006.

MCCORDUCK, P. **Machines who think : a personal inquiry into the history and prospects of artificial intelligence**. Natick, Mass.: A.K. Peters, 2004. ISSN 1568812051 9781568812052.

MCCULLOCH, W. S.; PITTS, W. H. **A Logical Calculus of The Ideas Immanent in Nervous Activity**. [S.l.: s.n.], 1943. v. 5. 115-133 p.

MOGHADDAM, M. J.; HOSSEINI, M.; SAFABAKHSH, R. Traffic light control based on fuzzy q-leaming. In: **Proc. The Int. Symp. Artificial Intelligence and Signal Processing (AISP)**. [S.l.: s.n.], 2015. p. 124–128.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. **Foundations of Machine Learning**. 2nd. ed. [S.l.]: The MIT Press, 2018. ISBN 0262039400.

MORI, K.; CHRISTODOULOU, A. Review of sustainability indices and indicators: Towards a new city sustainability index (csi). **Environmental Impact Assessment Review**, Elsevier, v. 32, n. 1, p. 94–106, 2012.

MULDER STEVEN BETHARDB, M.-F. M. W. D. A survey on the application of recurrent neural networks to statistical language modeling. 2015.

MÜLLER, J.; DIETMAYER, K. Detecting traffic lights by single shot detection. In: IEEE. **2018 21st International Conference on Intelligent Transportation Systems (ITSC)**. [S.l.], 2018. p. 266–273.

NAIR, V.; HINTON, G. Rectified linear units improve restricted boltzmann machines vinod nair. In: . [S.l.: s.n.], 2010. v. 27, p. 807–814.

NELLORE, K.; HANCKE, G. Traffic management for emergency vehicle priority based on visual sensing. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 16, n. 11, p. 1892, 2016.

NING, C. et al. Inception single shot multibox detector for object detection. In: . [S.l.]: IEEE, 2017. p. 549–554. ISBN 978-1-5386-0561-5.

NWANKPA, C. et al. **Activation Functions: Comparison of trends in Practice and Research for Deep Learning**. 2018.

PENG, Q. et al. Pedestrian detection for transformer substation based on gaussian mixture model and yolo. In: . [S.l.]: IEEE, 2016. v. 02, p. 562–565. ISBN 978-1-5090-0769-1.

PON, A. et al. A hierarchical deep architecture and mini-batch selection method for joint traffic sign and light detection. In: IEEE. **2018 15th Conference on Computer and Robot Vision (CRV)**. [S.l.], 2018. p. 102–109.

RAMACHANDRAN, P.; ZOPH, B.; LE, Q. V. Searching for activation functions. **CoRR**, abs/1710.05941, 2017. Disponível em: <http://arxiv.org/abs/1710.05941>.

RAZAVIAN, A. S. et al. Cnn features off-the-shelf: An astounding baseline for recognition. In: **2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops**. [S.l.: s.n.], 2014. p. 512–519.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 779–788.

REDMON, J.; FARHADI, A. Yolo9000: Better, faster, stronger. In: . [S.l.]: IEEE, 2017. p. 6517–6525. ISBN 978-1-5386-0458-8. ISSN 1063-6919.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. arxiv 2018. **arXiv preprint arXiv:1804.02767**, 2019.

REN KAIMING HE, R. G. S.; SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. 2016.

REN, P.; FANG, W.; DJAHEL, S. A novel yolo-based real-time people counting approach. In: . [S.l.]: IEEE, 2017. p. 1–2. ISBN 978-1-5386-2525-5.

ROTHAUS, K.; ROTERS, J.; JIANG, X. Localization of pedestrian lights on mobile devices. In: ASIA-PACIFIC SIGNAL AND INFORMATION PROCESSING ASSOCIATION, 2009 ANNUAL .... **Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference**. [S.l.], 2009. p. 398–405.

RUSSAKOVSKY, O. et al. Imagenet large scale visual recognition challenge. **International Journal of Computer Vision**, v. 115, 09 2014.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 4th ed.. ed. [S.l.: s.n.], 2020. ISBN 0-13-461099-7.

SAZLI, M. A brief review of feed-forward neural networks. v. 50, p. 11–17, 01 2006.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural Networks**, v. 61, 04 2015.

SILVA, C. M.; AQUINO, A. L.; MEIRA, W. Smart traffic light for low traffic conditions. **Mobile networks and applications**, Springer, v. 20, n. 2, p. 285–293, 2015.

SILVAA, M. K. e. K. H. B. N. Towards sustainable smart cities: A review of trends, architectures,components, and open challenges in smart cities. 2018.

SMITH CHRIS; MCGUIRE, B. H. T.; YANG, G. The history of artificial intelligence. **University of Washington**, 2006.

SMOLA, A.; VISHWANATHAN, S. **Introduction to Machine Learning**. [S.l.: s.n.], 2008.

SPERLING, D.; GORDON, D. Two billion cars: transforming a culture. **TR news**, n. 259, 2008.

SZEGEDY, C. et al. Going deeper with convolutions. In: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. p. 1–9.

TAHMID, T.; HOSSAIN, E. Density based smart traffic control system using canny edge detection algorithm for congregating traffic information. In: . [S.l.]: IEEE, 2017. p. 1–5. ISBN 978-1-5386-2308-4.

TAO, J. et al. An object detection system based on yolo in traffic scene. In: . [S.l.]: IEEE, 2017. p. 315–319. ISBN 978-1-5386-0494-6.

TAO, J. et al. An object detection system based on yolo in traffic scene. In: **2017 6th International Conference on Computer Science and Network Technology (ICCSNT)**. [S.l.: s.n.], 2017. p. 315–319.

THARW, A. Classification assessment methods. 2018.

THARWAT, A. Classification assessment methods. **ScienceDirect**, 2018.

TOSCANO, W. Inteligência artiflcial introdução. **UNINOVE**, 2009.

UIJLINGS, J. R. R. et al. Selective search for object recognition. **International Journal of Computer Vision**, v. 104, n. 2, p. 154–171, 2013. ISSN 1573-1405. Disponível em: <https://doi.org/10.1007/s11263-013-0620-5>.

VALIATI, G. R.; MENOTTI, D. Detecting pedestrians with yolov3 and semantic segmentation infusion. In: . [S.l.]: IEEE, 2019. p. 95–100. ISBN 978-1-7281-3228-0. ISSN 2157-8672.

VALUEVA, M. et al. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. **Mathematics and Computers in Simulation**, Elsevier, 2020.

WHO, U.; MATHERS, C. et al. Global strategy for women's, children's and adolescents' health (2016-2030). **Organization**, v. 2016, n. 9, 2017.

WU, Y.; TSAI, C. Pedestrian, bike, motorcycle, and vehicle classification via deep learning: Deep belief network and small training set. In: **Proc. Int. Conf. Applied System Innovation (ICASI)**. [S.l.: s.n.], 2016. p. 1–4.

WU, Z. et al. Multi-scale vehicle detection for foreground-background class imbalance with improved yolov2. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 19, n. 15, p. 3336, 2019.

XU, B. et al. Empirical evaluation of rectified activations in convolutional network. **arXiv preprint arXiv:1505.00853**, 2015.

YANG, Z. et al. Combining yolov3-tiny model with dropblock for tiny-face detection. In: IEEE. **2019 IEEE 19th International Conference on Communication Technology (ICCT)**. [S.l.], 2019. p. 1673–1677.

ZAID, A. A.; SUHWEIL, Y.; YAMAN, M. A. Smart controlling for traffic light time. In: **Proc. IEEE Jordan Conf. Applied Electrical Engineering and Computing Technologies (AEECT)**. [S.l.: s.n.], 2017. p. 1–5.

ZHANG, X.; ZHU, X. Vehicle detection in the aerial infrared images via an improved yolov3 network. In: IEEE. **2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP)**. [S.l.], 2019. p. 372–376.

ZHANG, Y. **New Advances in Machine Learning**. [S.l.]: InTech (February 01, 2010), 2010. ISBN 978-953-51-5906-3.

ZHAO, C.; CHEN, B. Real-time pedestrian detection based on improved yolo model. In: . [S.l.]: IEEE, 2019. v. 2, p. 25–28. ISBN 978-1-7281-1860-4.

ZHAO, Z. et al. Object detection with deep learning: A review. **IEEE Transactions on Neural Networks and Learning Systems**, v. 30, n. 11, p. 3212–3232, 2019.

ZHAO, Z.-Q. et al. Object detection with deep learning: A review. IEEE, v. 30, p. 3212–3232, 2019. ISSN 2162-2388.

ZHENG, Y. et al. A method of detect traffic police in complex scenes. In: . [S.l.]: IEEE, 2018. p. 83–87. ISBN 978-1-7281-0170-5.

ZHOU, Y. et al. **Soft-Root-Sign Activation Function**. 2020.

ZOU, X. A review of object detection techniques. In: **2019 International Conference on Smart Grid and Electrical Automation (ICSGEA)**. [S.l.: s.n.], 2019. p. 251–254.