

UNIOESTE - UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ  
CENTRO DE ENGENHARIAS E CIÊNCIAS EXATAS  
CURSO DE CIÊNCIA DA COMPUTAÇÃO

## **Rede Neural Convolucional para detecção de pedestres realizando travessias de risco**

Milena Lucas dos Santos

FOZ DO IGUAÇU

2022

Milena Lucas dos Santos

## **Rede Neural Convolucional para detecção de pedestres realizando travessias de risco**

Monografia submetida à Universidade Estadual do Oeste do Paraná, Curso de Ciência da Computação - Campus de Foz do Iguaçu, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Dr<sup>a</sup> Fabiana Frata Furlan Peres

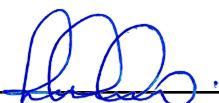
Valéria Nunes dos Santos

FOZ DO IGUAÇU

2022

Milena Lucas dos Santos

**Rede Neural Convolucional para detecção de pedestres  
realizando travessias de risco**



---

Drª Fabiana Frata Furlan Peres  
Orientador(a)



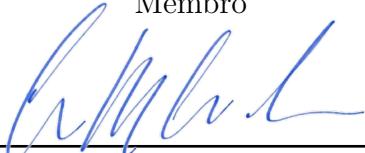
---

Valéria Nunes dos Santos  
Coorientador(a)



---

Drº Newton Spolaôr  
Membro



---

Drº Antônio Marcos M. Hachisuca  
Membro

*Dedico este trabalho à minha família.*

# Agradecimentos

Primeiramente, meus agradecimentos são a Deus, que tem me ajudado a chegar até aqui. Minha família que sempre me apoiou, em especial minha mãe.

Agradeço a professora e orientadora Fabiana, que tem me acompanhado desde o começo da minha graduação. Agradeço a Valéria, minha co-orientadora que sempre tem me ajudado e me ensinado pacientemente.

Agradeço aos demais professores, professor Cláudio, Eliane, Newton e todos os demais que me ajudaram na minha formação.

Agradeço aos meus colegas da faculdade, Raianny, Gabriel, Kame, Levi, Lucas, Joana, Victor, Marco e todos os demais que me apoiaram e me incentivaram sempre.

Sou grata à Unioeste e ao PTI por toda a experiência e aprendizado que passei ao longo de todos esses anos.

*“Os céus e a terra passarão, mas as minhas palavras jamais passarão.”*  
*(Mateus 24:35)*

# Resumo

Indivíduos atuam como pedestres quando encontram-se andando ou correndo em uma via. As principais interações entre pedestres e veículos ocorrem nas travessias de pedestres. Essas interações expõem os pedestres ao risco de acidentes e atrasos nos deslocamentos. Os pedestres estão suscetíveis a ferimentos graves e lesões que levam à morte e incapacidade, alarmando a saúde pública e a segurança do tráfego para tomar providências para tornar os pedestres menos expostos às situações de riscos produzidas pelo trânsito. O objetivo deste trabalho é detectar pedestres realizando travessias de risco em uma via real utilizando imagens de vídeos transmitidos em tempo real. Para isso, as tecnologias utilizadas foram CNN Yolov4-tiny para detecção e algoritmo SORT para rastreamento e contagem dos pedestres. O modelo final obteve uma precisão (*precision*) aproximada de 89%. Em média, a inferência da aplicação levou de 11 a 13 *frames* por segundo.

**Palavras-chaves:** Pedestres, Rede Neural Convolucional, Yolo, Travessia de risco.

# Listas de ilustrações

Figura 1 – Viés de representação da (a) Árvore de Decisão e (b) RNA.	7
Figura 2 – Hierarquia de aprendizado.	8
Figura 3 – Exemplo básico das tarefas preditivas.	9
Figura 4 – Neurônio.	10
Figura 5 – Um perceptron com 3 neurônios de entrada e um neurônio de saída.	13
Figura 6 – O funcionamento do perceptron.	13
Figura 7 – Exemplo de conjuntos linearmente separáveis e não linearmente separáveis.	14
Figura 8 – Perceptron multicamadas.	14
Figura 9 – Os neurônios nas vários camadas da rede MLP desempenhando diferentes funções.	15
Figura 10 – Os componentes de uma camada de rede neural convolucional típica.	18
Figura 11 – Uma ilustração da entrada e saída de uma operação de <i>max pooling</i> usando um campo receptivo 2x2 com stride 2. 3/4 dos neurônios de entrada são descartados neste procedimento.	20
Figura 12 – Detecção de objetos usando YOLO. Ele cria uma grade s x s a partir da imagem e prevê <i>bounding boxes</i> B, um nível de confiança para essas caixas (B) e probabilidades da classe de objeto para cada célula da grade.	26
Figura 13 – A arquitetura do Yolo consiste em 24 camadas convolucionais.	27
Figura 14 – Estrutura da CNN Darknet-19.	28
Figura 15 – Representação simples de como o Yolov3 extrai características de cada célula usando três escalas distintas.	30
Figura 16 – Visão geral da arquitetura YOLOv4. Formado por quatro blocos, CSP Darknet53, SPP, PANet e <i>Detection Head</i> .	31
Figura 17 – Estrutura da rede YOLOv4-Tiny.	32
Figura 18 – Tempos climáticos diferentes registrados.	35
Figura 19 – Distorção na imagem.	35
Figura 20 – LabelImg.	36
Figura 21 – Função do módulo DNN.	37
Figura 22 – Validação Cruzada K-fold.	40
Figura 23 – Metodologia utilizada no desenvolvimento do trabalho.	41
Figura 24 – Exemplificando o cálculo da métrica IoU.	43
Figura 25 – Imagem do conjunto de teste que ilustra diversas pessoas na via.	46

Figura 26 – Imagem do conjunto de teste que ilustra diversas pessoas na via sendo detectadas pela CNN. . . . .	46
Figura 27 – Ciclista detectado pela CNN. . . . .	47
Figura 28 – Estranho FP que a CNN detectou. . . . .	47
Figura 29 – Gráfico que relaciona AP com a função de perda. . . . .	48
Figura 30 – Linhas amarelas traçadas para contagem de pedestres. . . . .	49
Figura 31 – Contagem de pedestre realizando travessia de risco. . . . .	50
Figura 32 – Contagem de pedestre realizando travessia fora da faixa de pedestre. . . . .	50
Figura 33 – Pedestre atravessando fora da faixa, mas a contagem não é feita. . . . .	51

# **Lista de tabelas**

Tabela 1 – Diferenças entre arquivos Yolov4-tiny.	39
Tabela 2 – Hiperparâmetros alterados nos treinamentos.	40
Tabela 3 – Resultados dos 5 folds.	45
Tabela 4 – Resultados da média e desvio padrão para as métricas AP, IoU e F1-Score.	45
Tabela 5 – Resultados do modelo final.	48

# Listas de Abreviaturas e Siglas

*OpenCV Open Source Computer Vision Library*

AM Aprendizado de Máquina

AP *Average Precision*

BoF *Bag of Freebies*

BoS *Bag of Specials*

Catve Cascavel TV Educativa

CIoU *Complete Interaction over Union*

CNN *Convolutional Neural Networks*

CSP *Cross-Stage-Partial-connection*

CTB Código de Trânsito Brasileiro

cuDNN *NVIDIA CUDA Deep Neural Network*

DIoU *Distance Intersection over Union*

DNN *Deep Neural Network*

FC *Fully Connected*

FN Falso negativo

FP Falso positivo

FPN *Feature Pyramid Networks*

FPS *Frames per second*

HTTP *Hypertext Transfer Protocol*

IA Inteligência Artificial

IoU *Intersection over Union*

mAP *Mean Average Precision*

MLP *Multilayer Perceptron*

PANet *Path Aggregation Network*

PR Precisão-Revocação

ReLU *Rectified Linear Unit*

RNA Rede Neural Artificial

SGP Solucionador de Problemas Gerais

SORT *Simple Online RealTime Tracking*

SPP *Spatial Pyramid Pooling*

SSD *Single Shot MultiBox Detector*

SVM *Support Vector Machine*

VP Verdadeiro positivo

YOLO *You Only Look Once*

# Sumário

<b>1</b>	<b>Introdução . . . . .</b>	<b>1</b>
1.1	Objetivo Geral . . . . .	2
1.2	Objetivos Específicos . . . . .	2
1.3	Organização do Trabalho . . . . .	2
<b>2</b>	<b>Revisão Bibliográfica . . . . .</b>	<b>4</b>
2.1	Inteligência Artificial . . . . .	4
2.2	Aprendizado de Máquina . . . . .	6
2.2.1	Aprendizado Supervisionado . . . . .	9
2.3	Rede Neural Artificial . . . . .	10
2.3.1	Perceptron . . . . .	12
2.3.2	Perceptron Multicamadas . . . . .	14
2.4	Rede Neural Convolucional . . . . .	17
2.4.1	Convolução . . . . .	18
2.4.2	<i>Pooling</i> . . . . .	20
2.4.3	ReLU . . . . .	21
2.4.4	Data Augmentation . . . . .	21
2.4.5	Transfer Learning . . . . .	22
2.4.6	Aplicações das CNNs . . . . .	22
2.4.6.1	Detecção de objetos . . . . .	23
2.4.6.2	Vídeos . . . . .	25
2.5	You Only Look Once . . . . .	26
2.5.1	Yolov2 . . . . .	27
2.5.2	Yolov3 . . . . .	28
2.5.3	YOLOv4 . . . . .	29
2.5.4	Yolov4-tiny . . . . .	31
2.6	Trabalhos Relacionados . . . . .	32
<b>3</b>	<b>Materiais e Métodos . . . . .</b>	<b>34</b>
3.1	<i>Dataset</i> . . . . .	34
3.2	Ferramentas . . . . .	35
3.2.1	Pré-processamento . . . . .	35
3.2.2	LabelImg . . . . .	36
3.2.3	Treinamento . . . . .	36
3.2.4	Rastreamento . . . . .	37

3.3	Métodos . . . . .	39
<b>4</b>	<b>Resultados e Discussões . . . . .</b>	<b>42</b>
4.1	Métricas de Avaliação . . . . .	42
4.2	Resultados . . . . .	44
<b>5</b>	<b>Conclusão . . . . .</b>	<b>52</b>
	<b>Referências . . . . .</b>	<b>53</b>

# 1 Introdução

Os usuários mais vulneráveis no trânsito são pedestres, ciclistas e motociclistas, os quais correspondem cerca da metade do valor total das mortes causadas pelo trânsito no mundo. Nos dias atuais, há estimativas que indicam que acidentes de trânsito serão a quinta maior causa de morte no mundo no ano de 2030 [FERNANDES; BOING, 2019]. Já o Brasil está em quinto lugar entre os países que ocorrem mais mortes no trânsito, somente atrás da Índia, China, Estados Unidos e Rússia [SOUZA et al., 2022].

Todos os indivíduos atuam como pedestres diante do cenário do trânsito em certo momento. O local onde ocorrem as principais interações entre pedestres e veículos são as travessias de pedestres pelas vias. Essas interações são responsáveis pela maior exposição ao risco de acidentes e atrasos nos deslocamentos. Pedestres estão mais susceptíveis a ferimentos graves e lesões que levam à morte e incapacidade, trazendo preocupação para a saúde pública e segurança do tráfego. Por isso, é importante que providências sejam tomadas para tornar os pedestres menos expostos às situações negativas produzidas pelo trânsito [RIBEIRO; CALHÁO, 2017].

Desde 1998 com a criação do Código de Trânsito Brasileiro (CTB), medidas vem sendo tomadas para diminuir os acidentes no Brasil, com leis e penalizações para o motorista que desobedecer as leis estabelecidas. Um exemplo é a conhecida “Lei seca”, aprovada pelo Congresso Brasileiro em 2008, com o objetivo de impor penalizações mais severas para o condutor que dirigir sob a influência do álcool. Em 2010, o Governo brasileiro se comprometeu a desenvolver o Projeto RS 10 (*Road Safety in Ten Countries*), onde 10 países foram convidados a participar por apresentarem quase metade de todas as mortes de acidente no mundo. O objetivo do projeto é promover intervenções voltadas para a redução de velocidade, aprimoramento da legislação, aumento do uso do cinto de segurança, aumento da fiscalização sobre ”beber e dirigir”, melhoria da infraestrutura viária, entre outros [FERNANDES; BOING, 2019].

Frente ao cenário atual de mortes por acidente de trânsito, diferentes trabalhos tem sido conduzidas para que a tecnologia atual possa auxiliar na prevenção e redução de acidentes de trânsito. Há muitas investigações recentemente que buscam prever acidentes de trânsito ou acidentes fatais, utilizando modelos preditivos lineares [CHAKRABORTY; MUKHERJEE; MITRA, 2019]. Há pesquisas que avaliam o comportamento dos pedestres antes de atravessar a rua e que estimam se a ação do pedestre apresenta uma situação de risco [ZHANG et al., 2020]. Outras pesquisas utilizam *Deep Learning* para avaliar estradas seguras e suas condições [JAN et al., 2018]. O avanço do Sistema de alerta de colisão em cruzamentos tem trazido melhores resultados que o sistema convencional

[ZHANG et al., 2021].

Devido à realidade de acidentes de trânsito presente no país, criar soluções e meios que possam prevenir acidentes ou reduzir o risco de ocorrência deles é essencial no contexto atual. Dessa forma, este trabalho investiga um trecho de via de trânsito real a fim de auxiliar as entidades competentes a identificar a necessidade de uma melhor sinalização, devido à fluxo diferenciado de pedestres atravessando no mesmo local. Caso essa necessidade seja atendida, espera-se diminuir o impacto dos custos sociais, econômicos e pessoais gerados por acidentes de trânsito no local.

Em vista dos argumentos apresentados, este trabalho objetiva detectar pedestres ao travessar uma via de trânsito, local em que estão em maior exposição ao risco de sofrer um acidente. Dessa forma, com a detecção e o rastreamento de objetos em tempo real, é possível ter o conhecimento do número de pedestres que fazem uma travessia perigosa, gerando informações que possam ser apresentadas para autoridades competentes para que medidas sejam tomadas.

## 1.1 Objetivo Geral

O objetivo deste trabalho é desenvolver uma aplicação que identifique o fluxo de pedestres fazendo travessias perigosas usando uma Rede Neural Convolucional.

## 1.2 Objetivos Específicos

Dentre os principais objetivos específicos destacam-se:

- Gerar informações sobre o número de pedestres que realizam essa travessia em um determinado período.
- Estudar e aplicar uma técnica de rastreamento nos pedestres em vídeos.

## 1.3 Organização do Trabalho

Os capítulos desta monografia estão organizados da seguinte forma:

- No Capítulo 2 são apresentados a fundamentação teórica, trabalhos relacionados e tecnologias utilizadas para o trabalho.
- No Capítulo 3 são definidos os materiais utilizados e o modo como foram utilizados.
- No Capítulo 4 são abordados os resultados alcançados e discussões sobre os mesmos.

- E, por fim, no Capítulo 5 são apresentadas as conclusões obtidas sobre o trabalho e trabalhos futuros.

## 2 Revisão Bibliográfica

### 2.1 Inteligência Artificial

A Inteligência Artificial (IA) é uma área muito recente dentro de ciências e engenharia. Sua origem veio após a Segunda Guerra Mundial e seu nome foi criado em 1956. Há diversas definições de IA, segundo Rich e Knight [1991], IA é o estudo de como os computadores podem fazer tarefas que hoje são melhor desempenhadas pelas pessoas. Já para Schalkoff [1990], um campo de estudo que busca explicar e emular comportamento inteligente em termos de processos computacionais [ROSA, 2011].

Os fundamentos de IA são formados por disciplinas como Filosofia, Matemática, Economia, Neurociência, Psicologia, Engenharia de Computadores, Teoria de controle e cibernética e Linguística. Essas disciplinas contribuíram com pontos de vistas, técnicas e ideias para a IA. [RUSSELL; NORVIG, 2013].

O primeiro trabalho reconhecido como IA foi o modelo de neurônios artificiais, no qual cada neurônio representava sendo “ligado” ou “desligado”. A mudança para “ligado” ocorria ao resultado da estimulação por meio de um número suficiente de neurônios vizinhos. Esse trabalho foi desenvolvido por Warren McCulloch e Walter Pitts [MCCULLOCH; PITTS, 1943], em 1943 e se fundamentaram em três aspectos, o aprendizado da fisiologia básica e da função dos neurônios no cérebro, a teoria da computação de Turing e um estudo formal da lógica proposicional criada por Russell e Whitehead [RUSSELL; NORVIG, 2013].

O SNARC, primeiro computador compatível com uma rede de neurônios artificiais, foi construído em 1950 por dois alunos de Harvard chamados de Marvin Minsky e Dean Edmonds. Esse computador utilizava três mil válvulas eletrônicas e um aparelho de piloto automático para simular uma rede de 40 neurônios [RUSSELL; NORVIG, 2013].

Os primeiros anos da IA foram cheios de sucessos, mas apenas de forma limitada. Quando se considera os computadores primitivos da época, os aparatos de programação disponíveis e a realidade de que os computadores eram, vistos como máquinas capazes de realizar operações aritméticas e nada mais; o fato de um computador poder realizar qualquer tarefa remotamente inteligente foi surpreendente [RUSSELL; NORVIG, 2013].

O Solucionador de Problemas Gerais (SPG) foi planejado desde o início para reproduzir códigos comportamentais humanos de resolução de problemas. Criado por Newell e Simon, observou-se que a maneira que o programa considerava as possíveis ações era semelhante com os dos seres humanos. Assim, o SPG pode ter sido o pioneiro a incorporar

o estudo de “pensar de forma humana”. De acordo com o teorema de convergência do perceptron [BLOCK; KNIGHT; ROSENBLATT, 1962], um algoritmo de aprendizagem pode alterar os pesos de conexão de um perceptron para corresponder a qualquer dado de entrada, desde que a correlação exista [RUSSELL; NORVIG, 2013].

Os pesquisadores de IA tinham uma confiança exagerada em seus estudos, devido ao excelente potencial dos sistemas inteligentes iniciais em casos simples. Contudo, a maioria desses primeiros sistemas falharam quando foram experimentados em agrupamentos de problemas mais longos ou em problemas mais difíceis [RUSSELL; NORVIG, 2013].

Posteriormente, esses sistemas foram chamados de métodos fracos, por não serem capazes de solucionar problemas grandes ou difíceis. O uso de informações mais amplas e específicas de domínio como alternativa aos métodos fracos permite maiores processos de raciocínio e facilita o tratamento de problemas que geralmente ocorrem em especialidades restritas [RUSSELL; NORVIG, 2013].

O programa DENDRAL [BUCHANAN; SUTHERLAND; FEIGENBAUM, 1969] foi um exemplo inicial dessa abordagem. O objetivo desse trabalho era resolver o problema e inferir a estrutura molecular a partir das informações fornecidas por um espectrômetro de massa. O DENDRAL foi significativo porque foi o primeiro sistema intensivo de conhecimento bem-sucedido, com um imenso número de regras de propósito específico subjacentes à sua capacidade.

Sistemas especialistas são sistemas que têm a capacidade de adquirir, armazenar e manipular conhecimento especializado de uma determinada área do conhecimento humano. R1 foi o primeiro sistema especialista comercial a ser próspero. Esse programa auxiliou na configuração de novos pedidos de sistemas de computadores na empresa Equipment Corporation [MCDERMOTT, 1982]. A empresa economizava cerca de 40 milhões de dólares por ano com o R1. Em 1988, diversos grupos de IA estavam produzindo ou examinando sistemas especialistas. A Du Pont, por exemplo, tinha 100 sistemas em uso e 500 em progresso. Como resultado, economizava cerca de 10 milhões de dólares por ano. Naquela época, a maioria das organizações mais importantes dos Estados Unidos tinham seu próprio grupo de IA que usava e analisava os sistemas especialistas [RUSSELL; NORVIG, 2013].

Por meio do avanço de IA na solução de problemas, pesquisadores voltaram o seu foco ao “agente como um todo”. O agente é qualquer entidade que reconheça seu ambiente por meio de sensores e aja sob o ambiente por meio de atuadores [RUSSELL; NORVIG, 2010]. A Internet é um dos ambientes mais importantes para agentes inteligentes, pois as aplicações Web se tornaram muito comuns. Sistemas de recomendação, mecanismos de pesquisa e agregadores de conteúdo de construção de sites utilizam as tecnologias de IA como base de implementação [RUSSELL; NORVIG, 2010].

Atualmente, há diversas aplicações de IA em diferentes áreas, por exemplo: Veículos robóticos, Reconhecimento de voz, Planejamento autônomo e escalonamento, Jogos, Combate a spam, Planejamento logístico, Robótica, Tradução automática, entre outros [ROSA, 2011].

## 2.2 Aprendizado de Máquina

Nos últimos anos, surgiu a necessidade de ferramentas computacionais mais aprimoradas, pelo fato do aumento da complexidade dos problemas trazidos para o âmbito computacional e pelo volume de dados acarretados por diversos setores. Para tal, são necessárias técnicas com pouca intervenção humana capazes de criar, a partir de exemplos passados, uma hipótese ou função, capaz de ter êxito na solução do problema tratado. O Aprendizado de Máquina (AM) é o processo de indução de uma hipótese a partir da experiência passada [FACELI et al., 2010].

A capacidade de aprendizagem é considerada necessária para o comportamento inteligente. Memorizar fatos, inspecionar e examinar as circunstâncias para adquirir fatos, praticar para melhorar as habilidades motoras/cognitivas e estruturar o novo conhecimento em representações apropriadas são todos aspectos das atividades relacionadas à aprendizagem. O campo de AM é focado em como construir programas de computador que melhoram automaticamente à medida que ganham experiência.

Na literatura, existem inúmeras definições de AM, como a que segue: “A capacidade de melhorar o desempenho na realização de alguma tarefa por meio da experiência” [MITCHELL et al., 1997]. AM visa criar abordagens de uso geral para extrair padrões importantes de dados, de forma ideal, sem a necessidade de conhecimento de domínio adicional. O aprendizado é um método de detecção automática de padrões e estrutura nos dados, ajustando os parâmetros do modelo [MITCHELL et al., 1997].

Para exemplificar, considere um conjunto de dados de pacientes hospitalares. Cada dado corresponde a um paciente e é uma tupla criada pelos valores das características ou atributos relacionados ao paciente, de maneira que descrevem a cada um. Os atributos, por exemplo, são nome, idade, sexo e sintomas de cada paciente.

Para algumas técnicas de AM, um dos atributos é considerado um atributo alvo e seus valores podem ser estimados usando os valores dos outros atributos, chamados de atributos de entrada ou preditores. O objetivo dessas técnicas é construir um modelo ou hipótese que possa relacionar os valores dos atributos de entrada de um dado ao valor de seu atributo de saída, aprendendo por meio de um subconjunto dos dados conhecido como conjunto de treinamento [FACELI et al., 2010].

Ao relacionar a abordagem anterior com o conjunto de dados citado, é possível

por exemplo induzir uma hipótese apta a fazer diagnósticos corretos para pacientes distintos daqueles que foram disponibilizados para aprender a regra de decisão. A hipótese deve apresentar uma capacidade de generalização que seja consideravelmente boa. Dessa maneira, será útil para dados que não estavam inclusos no conjunto de treinamento.

A explicação para a baixa generalização de uma hipótese, pode ser que ela esteja superajustando (*overfitting*) aos dados. A hipótese mostra-se memorizada ou concentrada nos dados de treinamento nesta situação. Por outro lado, o método AM pode gerar hipóteses com histórico ruim mesmo no subconjunto de treinamento, resultando em um subajustamento (*underfitting*). Isso pode acontecer quando as amostras de treinamento disponíveis não são representativas ou quando o modelo inferido é muito simplista, não conseguindo compreender os padrões de dados existentes [FACELI et al., 2010].

A indução é um tipo de raciocínio que consiste em afirmar uma verdade generalizada a partir da observação de alguns elementos. Para definir uma hipótese indutiva, cada algoritmo usa uma forma ou representação, chamado de viés (*bias*). Esse viés limita o conjunto de hipóteses que podem ser induzidas pelo algoritmo. Como exemplo, Árvores de Decisão usam uma estrutura de árvore, em que cada nó interno é representado por uma verificação sobre o valor de um atributo e cada nó externo está relacionado a uma classe. Redes Neurais Artificiais (RNA) descrevem uma hipótese pelos pesos das conexões da rede, relacionados a um conjunto de valores reais [FACELI et al., 2010]. Na Figura 1 apresenta-se o viés de representação associado aos métodos de Árvore de Decisão e RNA.



Figura 1 – Viés de representação da (a) Árvore de Decisão e (b) RNA.

Fonte: FACELI et al. [2010].

Adicionalmente, os algoritmos de AM contém um viés de busca. Esse viés busca procurar a hipótese que é mais adequada aos dados de treinamento, especificando como as hipóteses no espaço de hipóteses são encontradas. Sem viés não haveria aprendizado/generalização. Dessa forma, os modelos seriam especializados para os exemplos individuais [FACELI et al., 2010]. Por isso, os vieses são tão importantes para os algoritmos de AM.

Os algoritmos de AM são frequentemente empregados em uma ampla gama de tarefas que podem ser organizados usando uma variedade de critérios. Um dos critérios refere-se ao paradigma de aprendizagem a ser escolhido para resolver o problema [FACELI et al., 2010]. As tarefas de aprendizado podem ser divididas em: Preditivas e Descritivas.

- Preditivas: Encontra uma hipótese ou modelo, a partir do conjunto de treinamento com atributos de entrada e saída (atributo alvo). Esses algoritmos seguem o paradigma de aprendizado supervisionado (2.2.1).
- Descritivas: O objetivo é descrever ou explorar um conjunto de dados, não utilizando atributos de saída. Por isso, seguem o paradigma de aprendizado não supervisionado.

O termo “supervisionado” refere-se à presença de um “supervisor externo” familiarizado com o resultado esperado para cada ocorrência (atributos de entrada). Isso permite que o supervisor externo avalie a capacidade da hipótese induzida de prever o valor de saída para novos exemplos [FACELI et al., 2010].

O aprendizado não supervisionado inclui abordagens como agrupamento de dados, regras de associação e sumarização. A primeira tem como objetivo encontrar grupos de objetos semelhantes no conjunto de dados. Por sua vez, a segunda visa relacionar, por meio de regras, um grupo de atributos a outro grupo de atributos [FACELI et al., 2010]. Ou, ainda, encontrar uma descrição concisa e simples para um conjunto de dados.

Observa-se a hierarquia de tipos de paradigmas de aprendizado na Figura 2. No topo está o aprendizado indutivo. Em seguida, estão posicionados os tipos de aprendizado: supervisionado e não supervisionado. O rótulo de dados ou atributo alvo diferencia as tarefas supervisionadas, sendo que problemas de classificação e regressão contém, respectivamente, rótulos discretos e contínuos [FACELI et al., 2010].



Figura 2 – Hierarquia de aprendizado.

Fonte: FACELI et al. [2010].

Apesar dessa distinção básica entre modelos preditivos e descritivos, deve-se enfatizar que um modelo preditivo também pode fornecer uma descrição concisa de um conjunto de dados, enquanto um modelo descritivo pode fornecer previsões após ser validado.

A aprendizagem por reforço é um tipo de aprendizagem que não se enquadra em nenhuma das categorias listadas acima. O objetivo deste paradigma é reforçar ou premiar uma ação positiva enquanto pune uma ação ruim. Ensinar um robô a descobrir a trajetória

ideal entre dois lugares é uma ilustração de uma tarefa de reforço. A passagem por trechos pouco promissores é punida, enquanto a passagem por áreas promissoras é recompensada [FACELI et al., 2010].

### 2.2.1 Aprendizado Supervisionado

O presente trabalho dá maior ênfase no paradigma de aprendizado supervisionado, pois possui algoritmos de classificação em seu método. A tarefa de aprendizagem supervisionada é definida a partir de um conjunto de observações rotuladas de pares de treinamento  $\mathbf{D} = \{(x_i, f(x_i)), i = 1, \dots, n\}$ , em que  $f$  corresponde a uma função desconhecida. Um algoritmo de AM preditivo aprende uma aproximação  $\hat{f}$  da função desconhecida  $f$ . Essa função aproximada,  $\hat{f}$ , possibilita estimar o valor de  $f$  para novas exemplos [FACELI et al., 2010]. Conforme a natureza de  $f$ , é comum diferenciar duas possíveis circunstâncias:

- Classificação:  $y_i = f(x_i) \in \{C_1, \dots, C_m\}$ , ou seja,  $f(x_i)$  corresponde a valores em um conjunto discreto, não ordenado;
- Regressão:  $y_i = f(x_i) \in \mathbb{R}$ , ou seja,  $f(x_i)$  corresponde a valores em um conjunto infinito e ordenado de valores.

A Figura 3(a) descreve um cenário em que existem duas classes com somente dois atributos de entrada: os resultados de dois exames. O objetivo é encontrar um limite de decisão que divida os exemplos de uma classe dos exemplos da outra. O limite de decisão pode ser uma linha reta se os exemplos de uma classe forem linearmente separáveis dos exemplos da outra classe, pois os exemplos contêm dois atributos. Uma mistura de linhas é necessária se os casos não forem separáveis linearmente. Quando há mais de dois atributos de entrada nos exemplos, são utilizados hiperplanos de separação em vez de linhas retas [FACELI et al., 2010].

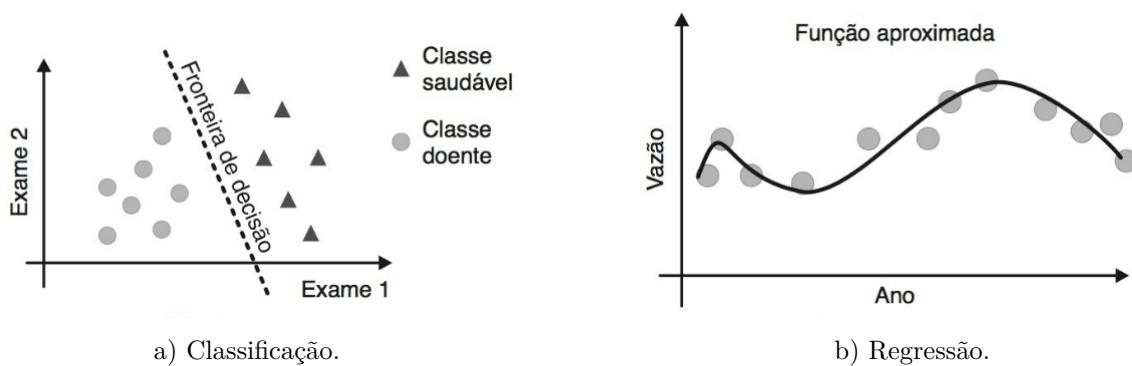


Figura 3 – Exemplo básico das tarefas preditivas.

Fonte: FACELI et al. [2010].

Vários métodos de AM podem apresentar diferentes fronteiras de decisão. Além disso, variações no conjunto de treinamento, na sequência na qual os exemplos são apresentados durante o treinamento e nos processos internos estocásticos podem fazer com que o mesmo algoritmo AM enfrente limites diferentes a cada treinamento subsequente [FACELI et al., 2010].

A Figura 3(b) descreve um problema de regressão em que o objetivo é aprender uma função que conecta um ano ao fluxo de água de um rio específico durante aquele ano. Apenas um atributo de entrada é usado nesta situação. Uma curva é produzida ligando os pontos gerados pelo dia e os valores de vazão. Esta curva deve se assemelhar à curva genuína da função que produz a vazão certa, dado o valor do ano. A regressão produzirá superfícies com mais de duas dimensões se mais de um atributo de entrada for fornecido [FACELI et al., 2010].

Apresenta-se um conjunto de testes de exemplos diferentes do conjunto de treinamento para medir a precisão de uma hipótese. Se uma hipótese prediz corretamente o valor de  $y$  para novos exemplos, pode-se considerar que generaliza bem. Quando a função  $f$  é estocástica — isto é, quando não é estritamente uma função de  $x$  — pode ser necessário aprender uma distribuição de probabilidade condicional,  $P(Y|x)$  [RUSSELL; NORVIG, 2013].

## 2.3 Rede Neural Artificial

As redes neurais artificiais (RNAs) são um tipo de abordagem de aprendizado de máquina que reproduz o mecanismo de aprendizado de organismos biológicos. Os neurônios, que estão interconectados no cérebro e envolvidos no processamento e transmissão de impulsos químicos e elétricos, compõem o sistema nervoso humano. Os neurônios são conectados uns aos outros com o uso de axônios e dendritos, e as regiões de conexão entre eles são chamadas de sinapses. Essas conexões são ilustradas na Figura 4. As forças das conexões sinápticas frequentemente mudam em resposta a estímulos externos. O aprendizado ocorre dessa maneira nos seres vivos [AGGARWAL, 2018; RASCHKA; MIRJALILI, 2017].

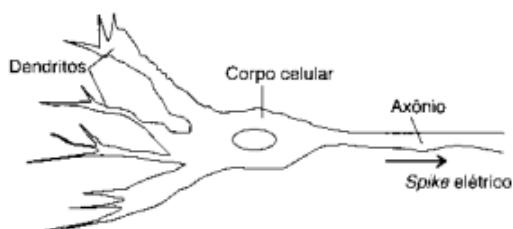


Figura 4 – Neurônio.

Fonte: Raschka e Mirjalili [2017].

Esse procedimento biológico é figurado em RNA, que possui unidades computacionais chamadas de neurônios também. Os pesos conectam unidades de computação umas às outras e servem ao mesmo propósito que as forças de conexão sináptica em seres biológicos. A entrada de cada neurônio é ponderada, o que tem impacto na função calculada naquela unidade. O neurônio pode ser descrito como uma porta lógica simples com saídas binárias. Muitos sinais chegam aos dendritos, são então integrados ao corpo celular e, se o sinal agregado exceder um determinado limiar, um sinal de saída específico é formado e enviado ao longo do axônio [AGGARWAL, 2018; RASCHKA; MIRJALILI, 2017].

Os estímulos externos são necessários para o aprendizado em indivíduos biológicos, assim também o estímulo externo nas RNAs é fornecido pelos dados de treinamento contendo exemplos de pares de entrada-saída da função a ser aprendida. Uma RNA computa uma função das entradas, utilizando os pesos como parâmetros centrais, disseminando os valores estimados dos neurônios de entrada para o(s) neurônio(s) de saída [AGGARWAL, 2018].

Os dados de treinamento entregam *feedback* dos pesos da rede neural indiretamente, por meio de medidas como a precisão. Os erros cometidos por uma RNA calculados por uma função estão relacionados, por exemplo, ao indivíduo que recebe um *feedback* ruim. Da mesma maneira, os pesos entre os neurônios são modificados em resposta às falhas de previsão. O objetivo de alterar os pesos é ajustar a função calculada para que as previsões subsequentes sejam mais precisas. Para diminuir a imprecisão neste caso, os pesos são minuciosamente modificados de uma maneira matematicamente razoável. A função computada pela rede neural é modificada ao longo do tempo, alterando gradualmente os pesos entre os neurônios em muitos pares de entrada-saída, resultando em previsões mais precisas [AGGARWAL, 2018].

As RNAs desenvolvem força combinando um grande número dessas unidades básicas e aprendendo os pesos das várias unidades juntas para reduzir o erro de previsão. A analogia biológica às vezes é vista como uma representação pobre do funcionamento do cérebro humano; ainda assim, os princípios da neurociência têm sido frequentemente eficazes no planejamento de arquiteturas de redes neurais [VASILEV et al., 2019]. É possível verificar dois aspectos principais em uma RNA:

- Arquitetura: especifica o tipo de conexão entre os neurônios, como *feedforward*, recorrente, multi ou de camada única, entre outros. Além disso, descreve o número de neurônios e de camadas.
- Aprendizagem: Define o tipo de treinamento. Os métodos mais comuns para o treinamento de uma RNA são o gradiente descendente e a retropropagação.

Uma RNA pode ter um número indeterminado de neurônios organizados em camadas que são interconectadas. O conjunto de dados e as circunstâncias iniciais são retratados na camada de entrada. Se a entrada for uma imagem em tons de cinza, por exemplo, a intensidade de um pixel da imagem representa a saída de cada neurônio nessa camada. Geralmente, a camada de entrada não é contada e sim a de saída, quando fala-se RNA de uma camada, está se referindo a uma camada de entrada e saída [AGGARWAL, 2018].

Por esse motivo, geralmente não contamos a camada de entrada como parte das outras camadas. Quando dizemos rede de 1 camada, na verdade queremos dizer que é uma rede simples com apenas uma camada, a saída, além da camada de entrada. Mais de um neurônio pode ser encontrado na camada de saída. Sobretudo, isso é útil na classificação, pois cada neurônio de saída representa uma classe diferente [AGGARWAL, 2018].

Os neurônios são organizados em camadas, pois um único neurônio pode gerar informações limitadas. Grupo de neurônios em várias camadas formam um vetor na camada de saída, por meio do cálculo de funções de ativação. Assim, é possível transmitir muito informações, não somente porque o vetor tem muitos valores, mas também porque as razões compatíveis entre eles trazem informações complementares [AGGARWAL, 2018]. Existem três elementos importantes para construir uma rede neural:

- Função de ativação: As entradas ponderadas e somadas são transformadas em um neurônio.
- Arquitetura: Como já citado anteriormente. O número de neurônios, o número de camadas e a maneira como os neurônios estão conectados.
- Algoritmo de treinamento: Como os pesos são determinados [DINOV, 2018].

### 2.3.1 Perceptron

Um perceptron é a RNA mais básica, contendo apenas uma camada de entrada e um nó de saída. Frank Rosenblatt criou o perceptron em 1957. Pode-se dizer também que o perceptron é um exemplo de RNA *feedforward* simples de uma camada [VASILEV et al., 2019; AGGARWAL, 2018]. Na Figura 5 apresenta-se a estrutura geral do perceptron.

O neurônio é definido da seguinte forma:

$$y = f(\sum_i x_i w_i + b) \quad (2.1)$$

Inicialmente, calcula-se a soma ponderada das entradas e pesos  $x_i$  e  $w_i$ . Os dados de entrada são representados por  $x_i$ , que são valores numéricos. Os pesos  $w_i$  são valores

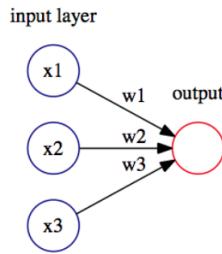


Figura 5 – Um perceptron com 3 neurônios de entrada e um neurônio de saída.

Fonte: Vasilev et al. [2019].

numéricos que representam a força das entradas ou, no caso dos neurônios, a força das conexões entre eles. O peso  $b$  é um valor único conhecido como viés e sua entrada é sempre 1 [VASILEV et al., 2019]. Vale ressaltar que o valor de  $y$  pertence ao intervalo [-1 a 1].

O resultado total da soma ponderada é então usado como entrada para a função de ativação  $f$ . A função de ativação limiar (*threshold*), principal função utilizada no perceptron, gera um sinal de saída assim que um limite de entrada especificado for atingido [VASILEV et al., 2019]. Na Equação 2.2 mostra-se a função limiar.

$$f(x) = \begin{cases} 0, & x < 0. \\ 1, & x \geq 0. \end{cases} \quad (2.2)$$

O resultado da função limiar é o resultado da saída do neurônio. Esse resultado é comparado com o valor esperado  $\hat{y}$ . Assim, o erro de previsão é  $E(x) = y - \hat{y}$ , que é um dos valores retirados do conjunto [-2, 0, +2], sendo  $x$  representado pelos valores de entrada. Os pesos da RNA devem ser atualizados na direção do gradiente de erro, quando o valor do erro  $E(x)$  for diferente de zero [AGGARWAL, 2018]. Na Figura 6 ilustra esse processo que ocorre no neurônio de saída.

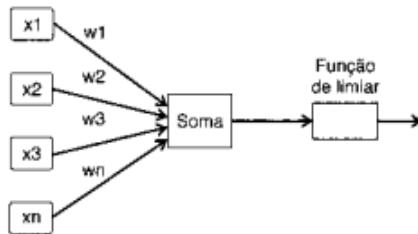


Figura 6 – O funcionamento do perceptron.

Fonte: Raschka e Mirjalili [2017].

Vale a pena ressaltar que a aproximação do perceptron só é confiável se as duas classes puderem ser separadas linearmente e a taxa de aprendizado for baixa consideravelmente. Se um limite de decisão linear não puder diferenciar as duas classes, define-se um número limite de passagens sobre o conjunto de dados de treinamento (épocas) e/ou

um limite no número de classificações errôneas aceitas, para que o perceptron conclua sua execução, deixando de atualizar os pesos [VASILEV et al., 2019]. Na Figura 7 exemplifica conjuntos separáveis de forma linear e não linear.

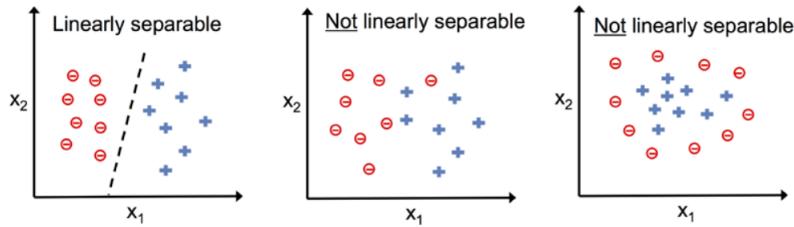


Figura 7 – Exemplo de conjuntos linearmente separáveis e não linearmente separáveis.

Fonte: Raschka e Mirjalili [2017].

### 2.3.2 Perceptron Multicamadas

O método mais comum para resolver problemas não linearmente separáveis com RNAs é adicionar uma ou mais camadas intermediárias [FACELI et al., 2010; VASILEV et al., 2019]. Uma rede com uma camada intermediária, segundo Cybenko [1989], pode implementar qualquer função contínua. Qualquer função pode ser aproximada usando duas camadas intermediárias. A Figura 8 mostra um exemplo da estrutura de um perceptron multicamadas.

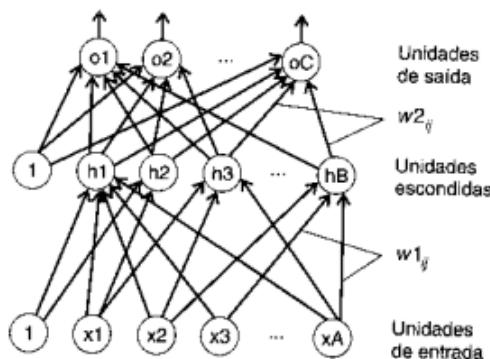


Figura 8 – Perceptron multicamadas.

Fonte: Aggarwal [2018].

Como os cálculos feitos por perceptron multicamadas ou *MultiLayer Perceptron* (MLP) não são visíveis para o usuário, as camadas intermediárias adicionais (entre entrada e saída) são chamadas de camadas ocultas. Como as camadas sucessivas alimentam umas às outras de maneira direta da entrada à saída, as MLPs têm uma arquitetura única conhecida como redes *feed-forward*. Todos os nós em uma camada são conectados aos da camada seguinte, de acordo com a arquitetura típica de redes *feed-forward* [AGGARWAL, 2018].

Nas camadas intermediárias de MLPs, são utilizadas funções de ativação não lineares, como a função sigmoidal. De fato, o aumento do poder de várias camadas requer a aplicação de funções de ativação não lineares [FACELI et al., 2010; AGGARWAL, 2018].

Cada neurônio em uma MLP tem um propósito distinto. Um neurônio em uma determinada camada desempenha uma função que é um composto de funções executadas pelos neurônios da camada anterior que estão conectados a ele. O processamento executado (e a função que o acompanha) ficam mais complexos medida que o processamento avança de uma camada intermediária para a próxima. Na primeira camada, cada neurônio aprende uma função que determina um hiperplano, que corta o espaço de entrada em duas seções. Cada neurônio na próxima camada forma regiões convexas combinando uma coleção de hiperplanos formados por neurônios na camada anterior. Um subconjunto das regiões convexas é combinado em regiões aleatórias pelos neurônios da próxima camada [FACELI et al., 2010].

A importância de cada neurônio na geração de limites de decisão que permitem à rede classificar novas amostras é exemplificada na Figura 9. A função associada à RNA como um todo é definida pela combinação de funções desempenhadas por cada neurônio da rede [FACELI et al., 2010].

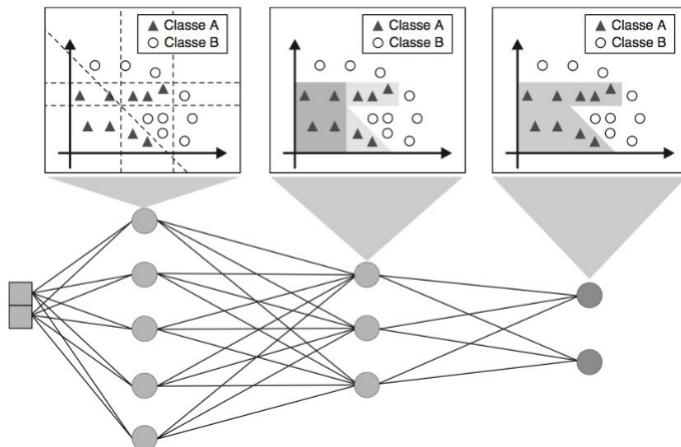


Figura 9 – Os neurônios nas vários camadas da rede MLP desempenhando diferentes funções.

Fonte: FACELI et al. [2010].

Um obstáculo que havia para utilizar redes multicamadas era a ausência de um algoritmo para o treinamento dessas redes, o que foi transposto com a proposta de um algoritmo de treinamento baseado em gradiente descendente denominado retropropagação. Para que esse algoritmo seja utilizado, a função de ativação precisa ser contínua, diferenciável e, de preferência, não decrescente [FACELI et al., 2010].

A função de ativação do tipo sigmoidal obedece a esses requisitos. Normalmente, todos os neurônios na mesma camada têm a mesma função de ativação, embora as funções de ativação em diferentes camadas podem ser distintas [AGGARWAL, 2018].

Cada objeto de entrada é fornecido à rede na fase *forward*. O objeto é primeiramente recebido por cada um dos neurônios da primeira camada intermediária da rede, onde é ponderado de acordo com o peso associado às suas respectivas conexões de entrada. Os neurônios dessa camada aplicam a função de ativação à sua entrada total e criam um valor de saída que é usado como valor de entrada pelos neurônios da próxima camada. Esse processo é repetido até que cada um dos neurônios na camada de saída produza um valor de saída, que é então comparado ao valor de saída desejado do neurônio. A diferença entre os valores de saída produzidos e requeridos de cada neurônio na camada de saída representa o erro da rede para o objeto apresentado [AGGARWAL, 2018].

Na fase de *backward*, o valor de erro de cada neurônio na camada de saída é usado para alterar seus pesos de entrada. Da camada de saída para a primeira camada intermediária, os ajustes são feitos. O algoritmo de retropropagação ajusta os pesos de uma rede MLP, conforme mostrado na Equação 2.3.

$$w_{jl}(t+1) = w_{jl}(t) + \eta x^j \delta_l \quad (2.3)$$

Nesta equação,  $w_{jt}$  denota o peso da conexão entre um neurônio  $l$  e o  $j$ -ésimo atributo de entrada ou saída do  $j$ -ésimo neurônio da camada anterior,  $\delta_l$  o erro associado ao  $l$ -ésimo neurônio e  $x^j$  a entrada recebida por esse neurônio (o  $j$ -ésimo sinal transmitido da camada anterior) [FACELI et al., 2010].

Como os valores de erro dos neurônios da camada de saída são conhecidos, o erro dos neurônios da camada intermediária deve ser calculado. A técnica de retropropagação apresenta um método para estimar o erro dos neurônios da camada intermediária com base nos erros detectados nos neurônios da camada posterior. A soma dos erros dos neurônios da próxima camada cujos terminais de entrada estão conectados a ela, ponderada pelo valor do peso associado a essas conexões, é utilizada para estimar o erro de um neurônio em uma determinada camada intermediária. Conforme demonstrado na Equação 2.4, o método de cálculo do erro depende da camada na qual o neurônio está colocado [FACELI et al., 2010].

$$\delta_l = \begin{cases} f'_a e_l & \text{se } n_l \in c_{sai} \\ f'_a \sum w_{lk} \delta_k, & \text{se } n_l \in c_{int} \end{cases} \quad (2.4)$$

Na Equação 2.4,  $n_l$  é o  $l$ -ésimo neurônio,  $c_{sai}$  é a camada de saída,  $c_{int}$  é uma camada intermediária,  $f'_a$  é a derivada parcial da função de ativação do neurônio e  $e_l$  é o erro quadrado do neurônio de saída quando sua resposta é comparada com a pretendida, que é especificado pela Equação 2.5.

$$e_l = \frac{1}{2} \sum_{q=1}^k (y_q - \hat{f}_q)^2 \quad (2.5)$$

Usando o gradiente descendente da função de ativação, a derivada parcial define a modificação dos pesos. A contribuição de cada peso no erro da MLP para a classificação de um objeto específico  $x$  é medida por esta derivada. Se esta derivada for positiva para um determinado peso, o peso está aumentando a diferença entre a saída da rede e a saída desejada. Como resultado, sua magnitude deve ser reduzida para reduzir o erro. Se a derivada for negativa, o peso está influenciando a saída da rede a ficar mais próxima do resultado desejado. Como resultado, seu valor deve ser aumentado [FACELI et al., 2010].

A retropropagação itera os ciclos de apresentação dos dados de treinamento e eventuais ajustes de peso até que um critério de parada seja alcançado. Diferentes condições de terminação, como um número máximo de ciclos ou uma taxa máxima de erro, podem ser utilizadas [FACELI et al., 2010].

O algoritmo de retropropagação é criticado por sua lenta convergência para um bom conjunto de pesos e baixo desempenho quando usado para grandes conjuntos de dados e tarefas difíceis. Mesmo para questões básicas, o conjunto de treinamento pode precisar ser apresentado centenas ou milhares de vezes, limitando sua aplicação a redes pequenas com alguns milhares de pesos [FACELI et al., 2010].

## 2.4 Rede Neural Convolucional

As redes convolucionais, comumente chamadas de *Convolutional Neural Networks* (CNN), são uma classe particular de rede neural usada para processar dados com uma arquitetura grade espacial [LECUN et al., 1989]. O termo “rede neural convolucional” refere-se a uma rede que usa a técnica matemática de convolução. Um tipo avançado de operação linear é a convolução. As redes convolucionais são essencialmente redes neurais com pelo menos uma camada que utiliza convolução em vez de multiplicação de matriz padrão [GOODFELLOW; BENGIO; COURVILLE, 2016].

Projetadas para se assemelhar ao córtex visual de um animal, essas redes empregam uma arquitetura singular que é especialmente eficaz na classificação de imagens. Os modelos mais utilizados para processamento de imagens e visão computacional são as CNNs [AGGARWAL, 2018].

As camadas da rede convolucional são cada uma composta por uma grade tridimensional com altura, largura e profundidade. Ao se referir a uma única camada, o termo “profundidade” refere-se ao número de canais em cada camada, como o número de canais de cores primárias (azul, verde e vermelho) na imagem de entrada ou a quantidade de mapas de recursos presentes nas camadas ocultas [AGGARWAL, 2018].

Os parâmetros de uma rede neural convolucional são organizados em grupos de filtros ou *kernel*, que são unidades estruturais tridimensionais. Filtros são matrizes qua-

dradas de dimensões  $n_k \times n_k$ , onde  $n_k$  é um número inteiro e frequentemente pequeno, como 3 ou 5. A profundidade de um filtro, por outro lado, é sempre a mesma da camada na qual é aplicado. O filtro pode ser movido em várias direções. *Stride* é o nome do parâmetro desta camada convolucional. Normalmente, o *stride* é constante em todas as dimensões de entrada [AGGARWAL, 2018].

Convolução, *pooling* e *Rectified Linear Unit* (ReLU) são os três tipos de camadas que são frequentemente encontrados em uma rede neural convolucional. Geralmente, uma CNN tem três estágios. A camada executa várias convoluções simultaneamente no primeiro estágio para criar uma coleção de ativações lineares. No segundo estágio, também conhecido como o estágio do detector, uma função de ativação não linear, como a função de ativação linear retificada ou ReLU, é aplicada a cada ativação linear. Altera-se ainda mais a saída da camada no terceiro estágio usando um mecanismo de *pooling* [GOODFELLOW; BENGIO; COURVILLE, 2016]. A Figura 10 apresenta esses estágios em uma camada convolucional.

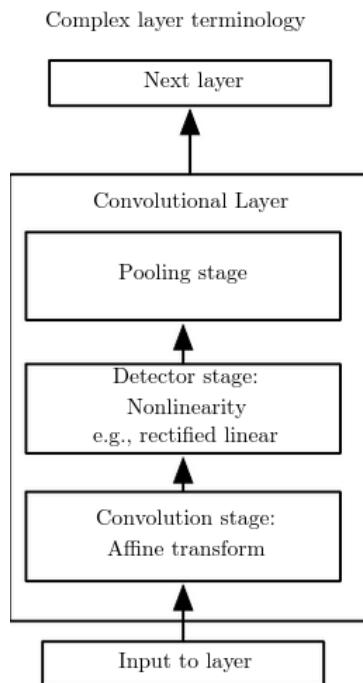


Figura 10 – Os componentes de uma camada de rede neural convolucional típica.

Fonte: Adaptada de Goodfellow, Bengio e Courville [2016].

### 2.4.1 Convolução

A convolução faz uso de representações equivariantes, interações esparsas e compartilhamento de parâmetros, três conceitos-chave que podem aprimorar um sistema de aprendizado de máquina [GOODFELLOW; BENGIO; COURVILLE, 2016].

- 1. Interações esparsas:** As camadas de uma rede neural tradicional multiplicam uma matriz de parâmetros com cada parâmetro indicando como cada unidade de entrada e cada unidade de saída interagem. Isso implica que cada unidade de entrada e unidade de saída se comunicam entre si. Por outro lado, interações esparsas são predominantes em redes convolucionais, sendo alcançadas ao se reduzir o tamanho do *kernel* em relação à entrada.

Ao processar uma imagem, por exemplo, a imagem de entrada pode ter milhares ou milhões de pixels, mas pode-se detectar recursos minúsculos e importantes, como bordas com *kernels* que ocupam apenas algumas dezenas ou centenas de pixels. Como resultado, as necessidades de memória do modelo são reduzidas e sua eficácia estatística é aumentada, exigindo que menos parâmetros sejam armazenados. Além disso, menos operações são necessárias para calcular o resultado. Esses ganhos de eficiência são bastante significativos [GOODFELLOW; BENGIO; COURVILLE, 2016].

- 2. Compartilhamento de parâmetros:** A prática de usar o mesmo parâmetro para muitas funções dentro de um modelo é conhecida como compartilhamento de parâmetros. Cada componente da matriz de pesos é usado exatamente uma vez ao calcular a saída de uma camada em uma rede neural convencional. É multiplicado por um elemento da entrada e nunca mais revisitado. Uma rede possui pesos vinculados, quando o valor do peso aplicado a uma entrada depende do valor de outro peso aplicado em outro lugar. Cada componente do *kernel* é utilizado em cada ponto da entrada em uma rede neural convolucional.

Como a operação de convolução compartilha parâmetros, precisa-se aprender apenas um conjunto de parâmetros geral, em vez de muitos conjuntos para cada local. O tempo de execução da propagação direta não é afetado por isso. A convolução é, portanto, muito mais eficiente em memória e estatisticamente do que a multiplicação de matrizes densas [GOODFELLOW; BENGIO; COURVILLE, 2016].

- 3. Representações equivariantes:** A convolução produz uma camada com uma propriedade conhecida como equivariância à tradução devido ao tipo específico de compartilhamento de parâmetros. Diz-se que uma função é equivariante se a saída varia da mesma forma que a entrada.

Nas imagens, a convolução constrói um mapa 2-D das localizações de recursos específicos na entrada. Mover um objeto na entrada fará com que sua representação se move na saída na mesma quantidade. Isso é útil quando sabe-se que uma função específica de um pequeno grupo de pixels próximos pode ser aplicada a vários locais de entrada. Por exemplo, detectar bordas na primeira camada de uma rede convolucional é útil para analisar imagens. É possível usar os mesmos parâmetros

em toda a imagem porque as mesmas bordas aparecem essencialmente em todos os lugares da imagem [GOODFELLOW; BENGIO; COURVILLE, 2016].

### 2.4.2 Pooling

Normalmente, as camadas de *pooling* são aplicadas logo após as camadas convolucionais. As informações na saída da camada convolucional são simplificadas pelas camadas de *pooling* [NIELSEN, 2015].

Uma camada de *pooling* cria um mapa de características condensado usando cada saída de mapa de características da camada convolucional. Por exemplo, cada unidade na camada de *pooling* pode totalizar, na camada anterior, uma região de 2x2 neurônios. Como exemplo, o termo “*max-pooling*” refere-se a um método frequente de *pooling*. No *max-pooling*, uma unidade de *pooling* produz apenas a ativação mais alta na região de 2x2 entradas [NIELSEN, 2015].

O processo de *pooling* é realizado no nível de cada mapa de ativação, diferentemente dos processos de convolução. Uma operação de *pooling* opera individualmente em cada mapa de características para produzir outro mapa de características, em contraste com uma operação de convolução, que emprega todos os mapas de características simultaneamente em conjunto com um filtro para produzir um único valor de recurso. Consequentemente, o processo de *pooling* não tem efeito sobre a quantidade de mapas de características. Em outras palavras, a camada que foi criada usando o *pooling* tem a mesma profundidade que a camada que foi usada para o processo de *pooling* [AGGARWAL, 2018].

O método mais popular de *pooling* é o *max pooling*. A técnica de *max pooling* propaga apenas o valor do neurônio com o maior valor de ativação em cada campo receptivo local (célula de grade) [VASILEV et al., 2019]. A Figura 11 mostra uma ilustração do *max pooling* com um campo receptivo 2x2.

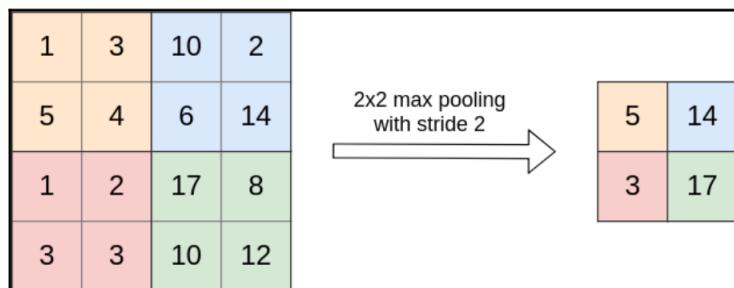


Figura 11 – Uma ilustração da entrada e saída de uma operação de *max pooling* usando um campo receptivo 2x2 com stride 2. 3/4 dos neurônios de entrada são descartados neste procedimento.

Fonte: Vasilev et al. [2019].

### 2.4.3 ReLU

As operações de *pooling* e ReLU são intercaladas entre a operação de convolução. A aplicação da ativação ReLU não é drasticamente diferente daquela de uma rede neural convencional. A função de ativação ReLU é aplicada a cada valor em uma camada para produzir valores de limiar. A camada subsequente recebe esses valores depois disso. Como resultado, como o ReLU é apenas um mapeamento direto de um para um dos valores de ativação, aplicá-lo não afeta as dimensões da camada. Nas redes neurais convencionais, a próxima camada de ativações é produzida combinando a função de ativação com uma transformação linear e uma matriz de pesos. Semelhante a isso, um ReLU geralmente vem após uma operação de convolução, que é o equivalente aproximado da transformação linear em redes neurais tradicionais [AGGARWAL, 2018].

É notável que a arquitetura de rede neural tenha avançado apenas recentemente para incluir a função de ativação ReLU. Funções de ativação saturantes como *sigmoid* e *tanh* foram utilizadas em épocas anteriores. O emprego do ReLU, no entanto, foi demonstrado em [Krizhevsky; Sutskever; Hinton, 2012] ter vantagens significativas de velocidade e precisão sobre essas funções de ativação. A precisão também está correlacionada com o aumento da velocidade, pois permite o uso de modelos mais profundos e tempos de treinamento mais longos [AGGARWAL, 2018].

### 2.4.4 Data Augmentation

O conceito de *data augmentation* (aumento de dados) é uma técnica predominante para reduzir o *overfitting* em redes neurais convolucionais. No aumento de dados, as transformações são aplicadas às instâncias originais para produzir novos exemplos de treinamento. Uma área em que o aumento de dados é especialmente adequado é o processamento de imagens. Isso ocorre porque muitas transformações, como translação, rotação, extração de *patch* e reflexão, não alteram realmente as características dos objetos em uma imagem. Quando usados no conjunto de dados suplementado, elas melhoram a capacidade de generalização do conjunto de dados. Por exemplo, o modelo é mais capaz de identificar bananas em várias orientações se um conjunto de dados de treinamento incluir imagens espelhadas e versões refletidas de todas as bananas nele [AGGARWAL, 2018].

O conjunto de imagens aumentado não precisa necessariamente ser expressamente preparada com antecedência para muitos desses tipos de aumento de dados porque envolvem muito pouca computação. O conjunto de dados e a aplicação em questão devem ser levados em consideração antes de aplicar o aumento de dados. A consideração mais importante ao determinar quais tipos de aumento de dados são justos é levar em conta o efeito de um tipo específico de aumento no conjunto de dados nos rótulos de classe, bem como a distribuição natural de imagens em todo o conjunto de dados.

### 2.4.5 Transfer Learning

A aprendizagem por transferência ou *transfer learning* é um método em que um modelo que foi treinado para resolver um problema específico é aplicado a um novo desafio associado ao problema original. Considere uma rede com várias camadas. No reconhecimento de imagem, as primeiras camadas geralmente captam padrões genéricos, enquanto as camadas posteriores captam padrões mais especializados [MICHELUCCI, 2019]. A seguir é descrito as etapas da aprendizagem por transferência:

- Etapa 1: Treina-se uma rede de base usando um conjunto de dados base relacionado ao problema ou adquire-se um modelo pré-treinado.
- Etapa 2: Adquire-se um novo conjunto de dados de treinamento, que pode ser chamado de conjunto de dados de destino. Na maioria das vezes, esse conjunto de dados será muito menor do que o utilizado na Etapa 1.
- Etapa 3: No conjunto de dados de destino, uma nova rede conhecida como rede de destino é treinada posteriormente. Às vezes, embora isso exija um hardware consideravelmente mais poderoso, é possível treinar toda a sua rede de destino usando os pesos que foram herdados da rede base como seus valores iniciais.

A rede básica aprendeu a extrair características gerais das imagens adequadamente na Etapa 1 e utiliza-se desse conhecimento em vez de ter que aprender tudo de novo. Mas, para melhorar suas previsões, adapta-se as previsões de sua rede à sua situação específica, maximizando como sua rede de destino extraí elementos específicos que são relevantes para seu problema, isso geralmente ocorre nas camadas finais da rede [MICHELUCCI, 2019].

O aprendizado de transferência é baseado na hipótese de que a Etapa 1 pode ser aprendida a partir de um grande número de imagens genéricas de uma rede básica e que a Etapa 3 pode ser aprendida a partir de um conjunto de dados muito menor usando o que foi aprendido na Etapa 1. Essa é uma ferramenta muito eficaz que ajudará a evitar o ajuste excessivo (*overfitting*) ao conjunto de dados de treinamento quando o conjunto de dados de destino for significativamente menor que o conjunto de dados base.

### 2.4.6 Aplicações das CNNs

As aplicações para redes neurais convolucionais incluem detecção de objetos, localização, vídeo e processamento de texto. Muitos desses aplicativos operam sob o princípio fundamental de que os recursos projetados são fornecidos por redes neurais convolucionais, sobre as quais aplicações multidimensionais podem ser construídos. Praticamente

nenhuma outra classe de redes neurais pode igualar o sucesso das redes neurais convolucionais [AGGARWAL, 2018]. A seguir, detecção de objetos e classificação de vídeos é descrito em mais detalhes.

#### 2.4.6.1 Detecção de objetos

A detecção de objetos é um subcampo da visão computacional e do processamento de imagens que permite o reconhecimento de objetos visuais em imagens e vídeos. O objetivo da detecção de objetos é criar modelos computacionais que digam aos aplicativos de visão computacional onde e que tipo de objeto eles estão olhando [ZOU et al., 2019].

A detecção de objetos é um dos métodos de IA que simula a visão humana e pode encontrar e identificar objetos. Além disso, possui aplicativos que permitem a detecção de objetos específicos, como detecção de pedestres e rostos [ZOU et al., 2019].

A detecção de objetos como técnica começou nos anos 2000 e pode ser dividida em duas fases: a fase tradicional, que não empregava aprendizado profundo, e a fase baseada em CNN. Algoritmos de detecção, que foram criados usando *features* projetadas à mão, eram uma marca registrada da era tradicional. O desenvolvimento de um modelo de detecção através da construção de representações de características complexas foi necessário porque não havia uma forma eficiente de representar a imagem [ZOU et al., 2019].

O desempenho das *features* projetadas à mão foi saturado durante esse período, e a Detecção de Objetos só pode avançar em 2014. O desenvolvimento foi viabilizado pelo ressurgimento das Redes Neurais Convolucionais, que possibilitou a extração de representações confiáveis e de alto nível de dados de recursos de uma imagem [ZOU et al., 2019]. A proposta de R. Girshick de Regiões com Recursos da CNN (R-CNN), que marcou o início da era da Detecção de Objetos, representou o primeiro grande avanço.

Tanto o conjunto de dados VOCs quanto o *benchmark* COCO são bem conhecidos no campo da detecção de objetos. Esses conjuntos de dados são empregados para avaliar a precisão dos modelos de detecção. Alguns deles atingem quantidades de dados de treinamento de 500.000 ou mais exemplos, com inúmeras classes de objetos [ZOU et al., 2019].

Os modelos de detecção de dois estágios e os modelos de detecção de um estágio, são os dois conjuntos de modelos de detecção usados no período dos algoritmos baseados em CNN [ZOU et al., 2019]. A seguir, são apresentados alguns modelos de detecção de dois estágios:

- R-CNN: Desenvolvido por Girshick et al. [2014], o modelo extrai um conjunto de propostas de regiões de imagem, dimensiona cada proposta para um tamanho fixo e alimenta os recursos para um modelo CNN treinado. Finalmente, um classificador

SVM linear é usado para prever se um objeto estará presente em cada região e para categorizar o objeto. No VOC07, o RCNN superou os 33,7% anteriores, atingindo um mAP (*Mean Average Precision*) de 58,5%.

- SPPNet: Um termo para as redes de agrupamento de pirâmides espaciais foi proposta por He et al. [2015] em 2014. A camada *Spatial Pyramid pooling*, que permite que uma CNN gere uma representação de tamanho fixo independente dos outros tamanhos, é a contribuição da SPPNet. A utilização do SPPNet permite que os mapas de recursos de toda a imagem sejam calculados apenas uma vez, economizando tempo ao eliminar a necessidade de calcular repetidamente os recursos convolucionais. O SPPNet atingiu um mAP VOC07 de 59,2% e foi 20 vezes mais rápido que o R-CNN.
- *Faster RCNN*: Girshick [2015] propôs isso em 2015. Foi o primeiro detector DL a detectar muito próximo do tempo real. A Rede de Propostas de Região, que permite uma proposta de região quase sem custo, contribuiu para o *Faster RCNN*. O *Faster RCNN* realizou a detecção a uma taxa de 17 quadros por segundo e teve mAPs de 73,2% no VOC07, 70,4% no VOC12 e 42,7% no COCO@.5.
- *Feature Pyramid Networks*: Baseado em *Faster RCNN*, Lin et al. [2017a] propôs o *Feature Pyramid Networks* (FPN) em 2017. A arquitetura é top-down e usa conexões laterais para criar semântica de alto nível. O FPN serviu como um componente de construção fundamental para detectores posteriores, pois foi capaz de obter 59,1% de mAP para COCO@.5.

A seguir, são apresentados alguns modelos de detecção de um estágio:

- *You Only Look Once* (YOLO) : Primeiro detector de estágio único, proposto por Redmon et al. [2016]. Com um mAP de 52,7%, o Yolo tem uma alta velocidade de processamento, atingindo 155 FPS (*Frames per second*) no VOC07. Esse algoritmo opera com o princípio de aplicar uma única rede neural à imagem. A rede divide a imagem em regiões e simultaneamente realiza a previsão da *bounding box* para cada região.
- O *Single Shot MultiBox Detector* (SSD) : O segundo detector de estágio único, foi proposto por Liu et al. [2016]. Ele também deu uma contribuição ao empregar várias referências e várias resoluções na detecção. Um SSD pode rodar a 59 quadros por segundo e obter mAPs de 76,8% no VOC07 (74,9% no VOC12) e 46,5% no COCO@.5.
- RetinaNet: Projetado para tentar aproximar a precisão dos detectores de 2 estágios, foi proposto em 2017 por Lin et al. [2017b]. Para fazer com que o detector se

concentre mais em amostras desafiadoras e mal classificadas durante o treinamento, uma nova função de perda denominada *focal loss* foi implementada. Ele reformula a *cross entropy loss* padrão. Os detectores de 1 estágio mantiveram sua alta velocidade de detecção com perda de foco e atingiram um mAP de 59,1% em COCO@.5.

#### 2.4.6.2 Vídeos

O problema de rastreamento de objetos em vídeo é significativo para a visão computacional. Sistemas de navegação de veículos, análise de vídeo esportivo, vigilância visual, prevenção de obstáculos e outros aplicativos o utilizam [DAWSON-HOWE, 2014]. Normalmente é difícil rastrear um objeto visualmente porque:

- Pode-se mover de forma complexa em relação à câmera.
- Pode-se alterar a forma.
- Pode-se ocasionalmente ser completamente ou parcialmente obstruído.
- A aparência pode ser alterada devido à iluminação ou ao clima.
- A aparência pode mudar fisicamente.

O rastreamento visual é, portanto, um tópico extremamente desafiador que tem sido tratado em uma variedade de métodos [DAWSON-HOWE, 2014]. Abaixo são apresentados alguns:

- *Exhaustive search*, onde uma imagem monitorada é pesquisada pela melhor correspondência em cada quadro.
- *Mean shift* usa um método gradiente de subida/descida para reduzir o número de comparações necessárias e uma representação de histograma do item para a comparação.
- *Optical flow* é um método para descobrir como todos os pixels (ou características esparsas) estão fluindo de um quadro para outro.
- Se restringirmos os pontos de características àqueles localizados dentro do objeto de interesse, o *Feature point tracking* pode ser utilizado para rastrear objetos. Isso pode oferecer um método mais rápido e confiável.
- Uma abordagem recursiva eficaz para estimar estados é o *Kalman filter*, que é uma coleção de equações matemáticas. É possível estimar com precisão uma variável não observável conhecida como “variável de estado” pela observação da variável conhecida como “variável de observação”. É possível estimar situações passadas,

atuais e até futuras. Para modelagem de pano de fundo, uma variedade de versões foram apresentadas, com a principal diferença sendo os espaços de estado usados para rastreamento [ANGHINONI, 2020].

## 2.5 You Only Look Once

Como sugere seu nome, esse algoritmo pode prever os itens em uma imagem e onde eles estarão, bastando olhar para ela apenas uma vez. A Figura 12 mostra a arquitetura do Yolo em geral. Em vez de usar uma janela deslizante para escanear a imagem inteira, essa arquitetura divide a imagem de entrada em uma grade de tamanho  $s \times s$ , com *bounding boxes* B em cada célula da grade. Essas *boxes* são responsáveis por calcular a probabilidade ( $P_r$ ) ou nível de confiança ( $P_r(\text{object}) \times IOU_{pred}^{truth}$ ) se houver um objeto nessa grade [REDMON et al., 2016].

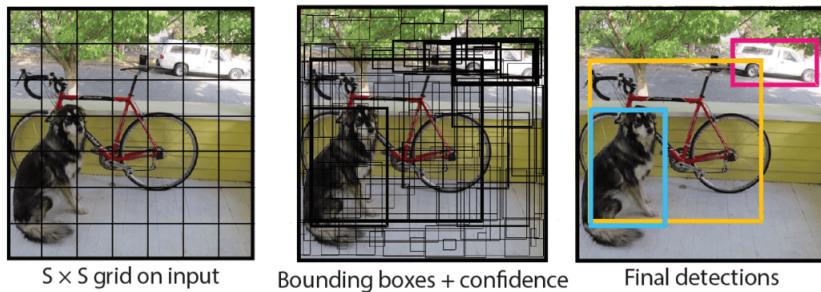


Figura 12 – Detecção de objetos usando YOLO. Ele cria uma grade  $s \times s$  a partir da imagem e prevê *bounding boxes* B, um nível de confiança para essas caixas (B) e probabilidades da classe de objeto para cada célula da grade.

Fonte: Redmon et al. [2016].

Embora as *bounding boxes* criadas não revelem informações sobre a classe de objetos, elas são mais aparentes em áreas onde há maior probabilidade de que um objeto exista (espesso). Cinco previsões formam *bounding boxes*: x, y, w, h e o nível de confiança, w e h são a largura e a altura normalizadas em relação ao tamanho geral da imagem, x e y denotam o centro da *bounding box* em relação aos limites da célula da grade. Finalmente, a probabilidade de que essa célula inclua um objeto é calculada usando a IoU (*Intersection over Union*) entre a *box* prevista, *ground truth* e o nível de confiança [REDMON et al., 2016].

A rede Yolo está inspirada na arquitetura de GoogLeNet, composta por 24 camadas convolucionais seguidas por duas camadas *Fully Connected* (FC) no final da rede. Além disso, o Yolo usa filtros com tamanhos de 3 x 3 para extrair recursos e filtros 1 x 1 para diminuir o número de canais na saída das camadas convolucionais no lugar dos módulos *inception* usados pelo GoogLeNet [REDMON et al., 2016]. Na Figura 13, a arquitetura Yolo é exibida.

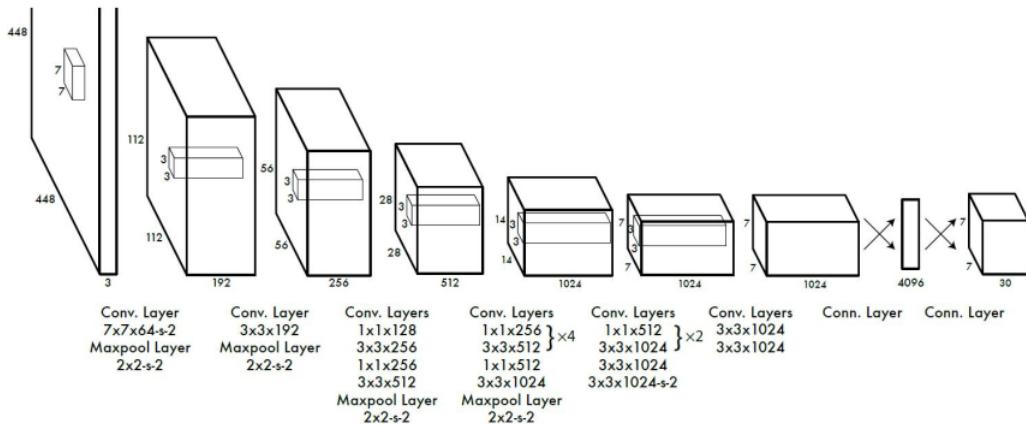


Figura 13 – A arquitetura do Yolo consiste em 24 camadas convolucionais.

Fonte: Redmon et al. [2016].

Até o momento, há quatro versões da Yolo, sendo Yolo, Yolov2, Yolov3 e Yolov4. A seguir, é descrito as versões 2, 3 e 4.

### 2.5.1 Yolov2

O objetivo da versão Yolov2 era preservar a velocidade de inferência em tempo real enquanto aumentava a precisão da detecção de objetos. Para fazer isso, eles alteraram significativamente o *design* Yolo, bem como a abordagem da fase de treinamento da rede. Embora CNN Yolo tenha sido substituído por um Darknet-19, que consiste em 19 camadas convolucionais e 5 camadas de *max pooling*, ele ainda opera na mesma premissa. Além disso, a *batch normalization* é empregada em todas as camadas convolucionais nesta versão como um método para evitar o *overfitting* [REDMON; FARHADI, 2017].

Uma das melhorias feitas pelo Yolov2 em sua estrutura foi o uso de *anchors boxes* para aprimorar a previsão da *bounding box*, semelhante à forma como o SSD e o *Faster R-CNN* o realizam. O YOLOv2 emprega a técnica de agrupamento K-means para determinar o tamanho ótimo das *anchors boxes* que é comparável ao tamanho real dos itens presentes nas imagens, diferentemente dessas duas técnicas onde as *anchors boxes* são determinadas manualmente. Redmon e Farhadi [2017] afirma que o emprego de cinco *anchors boxes* com proporções precisas permite um *trade-off* favorável entre a complexidade do modelo e o *recall*, auxiliando no aprendizado da rede [REDMON; FARHADI, 2017].

Dada a redução e/ou simplificação da quantidade de processos, a estrutura Darknet-19 utilizada no Yolov2 possibilitou uma melhora no desempenho do detector. Como seu antecessor, o YOLOv2 emprega 1 x 1 filtros para minimizar o número de canais na saída de convolução e 3 x 3 filtros para extrair mapas de características da maioria das camadas convolucionais. Nesse sentido, o Darknet-19 exige apenas 5,8 milhões de operações, em oposição aos 30,69 milhões da rede VGG16 e seu antecessor (Yolo), que exige 8,52

milhões. A Figura 14 mostra a arquitetura do Darknet-19. Os autores mudaram a rede excluindo as camadas FC encarregadas de prever *anchors boxes* e adicionando três camadas 3x3 convolucionais, cada uma com 1024 canais de saída. Em seguida, uma camada convolucional final de  $1 \times 1$  foi aplicada para reduzir o tamanho do tensor de  $7 \times 7 \times 1024$  para  $7 \times 7 \times 125$  [REDMON; FARHADI, 2017].

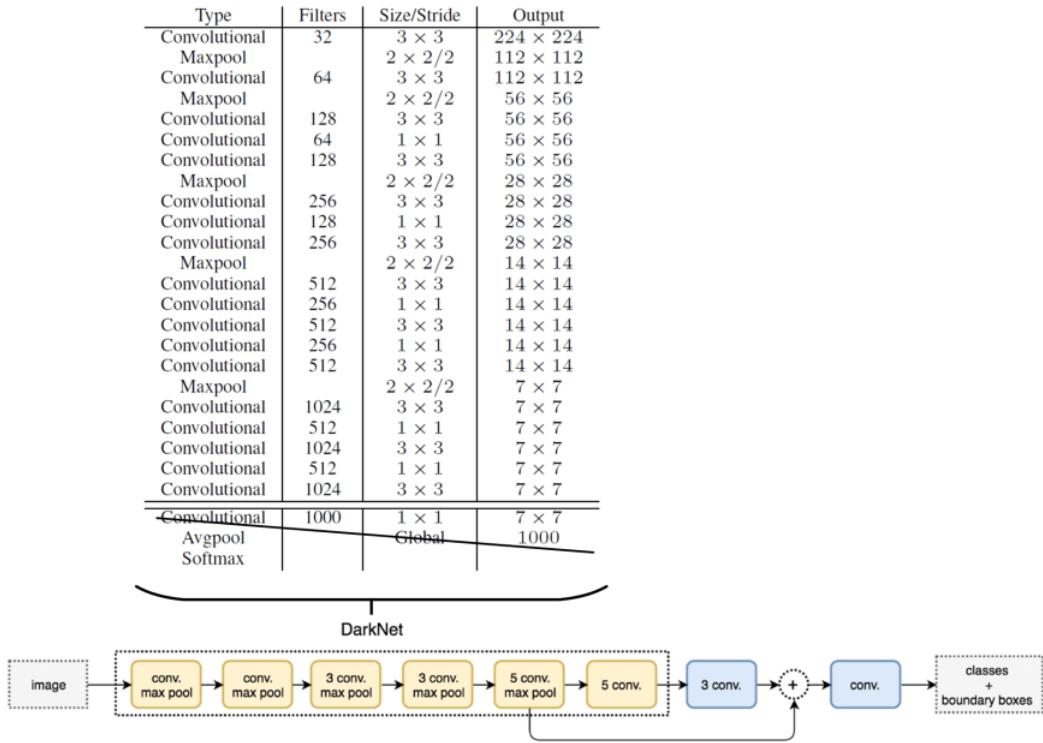


Figura 14 – Estrutura da CNN Darknet-19.

Fonte: Erazo et al. [2021].

A eliminação das camadas FC tem o benefício de permitir o treinamento multi-escala porque a rede pode acomodar vários tamanhos de imagem. O YOLOv2 seleciona aleatoriamente um novo tamanho de dimensão de imagem entre 320 x 320 pixels e 608 x 608 pixels a cada 10 *batches* (lotes) durante a fase de treinamento. A rede foi capaz de aprender uma ampla gama de dimensões de entrada e reconhecer objetos em várias escalas graças ao processo de treinamento [REDMON; FARHADI, 2017].

### 2.5.2 Yolov3

Yolov3 desenvolvida em 2018 usa uma CNN chamada Darknet-53 composta por 53 camadas inspirada em ResNet50 [HE et al., 2016]. A previsão de *bounding box*, previsão de classe e previsão de escala são três das técnicas mais importantes usadas pelo Yolov3 para aumentar a precisão da detecção [REDMON; FARHADI, 2018].

- Previsão de *bounding box*: YOLOv3 e YOLOv2 usam âncoras que possuem *boxes*

que se assemelham a objetos de vários tamanhos. Aqui, YOLOv3 emprega regressão logística para prever uma pontuação de objetividade (confiança) para cada *bounding box*. A pontuação de objetividade correspondente será 1 se a *bounding box* anterior se sobrepujar ao *ground truth* de um objeto mais do que qualquer outra *bounding box* anterior. Em contraste, se a *bounding box* se sobrepujar ao *ground truth* para um objeto, mas a sobreposição não exceder um limiar predeterminado ( $\text{IOU} \geq 0,5$ ), a sobreposição será desconsiderada. Dessa maneira, cada objeto do *ground truth* recebe uma única *bounding box* anterior [REDMON; FARHADI, 2018].

- Previsão de classe: A adição da classificação multi-rótulo a esta nova versão é uma mudança intrigante. Para resolver problemas com classificação multi-rótulo, YOLOv3 substitui classificadores logísticos independentes para a função *softmax*, pois o uso de uma camada *softmax* reforça a suposição irreal de que cada célula da grade tem exatamente uma classe. Ao empregar classificadores independentes, cada classe de objetos na imagem recebe um nível de probabilidade. Por exemplo, se a rede for treinada para reconhecer automóveis e caminhões, o modelo determina qual é a chance de haver um carro na imagem. Isso permite que um objeto seja detectado como classe 1 e classe 2 ao mesmo tempo, exatamente como calcula a probabilidade de ter um caminhão na imagem [REDMON; FARHADI, 2018].
- Predição através de escalas: O conceito FPN serviu de base para a terceira modificação (LIN et al., 2017a). Para trabalhar com imagens de várias resoluções e aprimorar a detecção de pequenos objetos, o YOLOv3 possui três *boxes* com três escalas diferentes para cada célula da grade nesta versão, ilustrada na Figura 15. A aplicação de  $N \times N$  núcleos de detecção com passo de 32, 16 e 8 é usada para realizar a detecção em mapas de características com três tamanhos diferentes em três pontos diferentes na rede. O volume do tensor de saída de cada previsão é  $N \times N \times 3 \times (4 + 1 + 80)$ , onde  $N$  é o tamanho da escala (por exemplo, 13 x 13, 26 x 26 e 52 x 52), 3 denota o número de *boxes*, 4 denota as coordenadas da *bounding box*, 1 denota a pontuação de objetividade e 80 denota o número de classes no COCO [REDMON; FARHADI, 2018].

### 2.5.3 YOLOv4

Um detector que se destacou no estado da arte é o modelo YOLOv4, que apresenta uma excelente combinação entre velocidade e precisão ( $\text{mAP}@[.5,.95]=43,5\%$ ) (65 FPS). A arquitetura YOLOv4 CSPDarknet-53 é baseada principalmente no Yolov3 Darknet-53, mas também incorpora novos blocos de convolução usados em detectores de um e dois estágios, como *Spatial Pyramid pooling* (SPP) [HE et al., 2015], *DenseNet* [HUANG et al., 2018], *Path Aggregation Network* (PANet) [LIU et al., 2018].

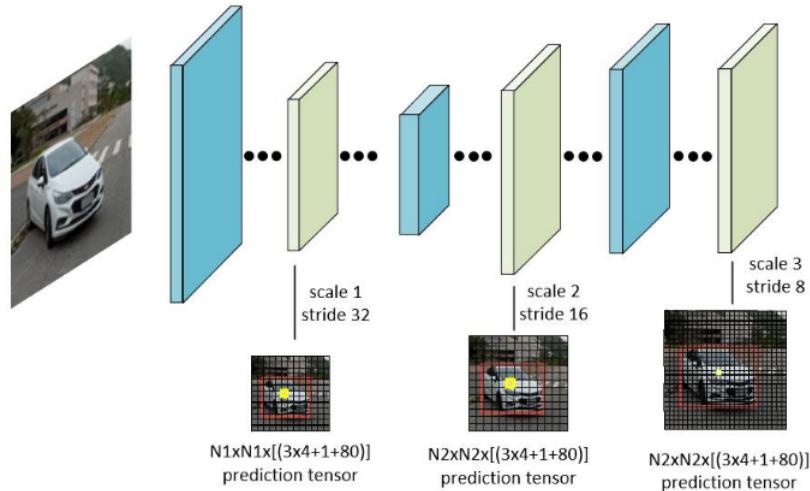


Figura 15 – Representação simples de como o Yolov3 extrai características de cada célula usando três escalas distintas.

Fonte: Erazo et al. [2021].

Conforme ilustrado na Figura 16, a arquitetura primária usada pelo YOLOv4 para extraírem mapas de características é CSPDarknet53. Para reduzir a complexidade computacional do modelo, a rede CSP (*Cross-Stage-Partial-connection*) é integrada ao Darknet53. O CSP permite que os mapas de características detectados sejam divididos em duas partes, com apenas metade através de uma camada de convolução espessa, reduzindo assim o número de parâmetros. Semelhante ao Yolov3, o Yolov4 cria uma estrutura hierárquica de mapas de características usando blocos adicionais para aprimorar a detecção de objetos em várias escalas. Para isso, o Yolov4 emprega o PANet como técnica de agregação de parâmetros em vários níveis do modelo e combina o bloco SPP para ampliar o campo receptivo [BOCHKOVSKIY; WANG; LIAO, 2020].

Ao combinar pirâmides espaciais, o bloco SPP permite a extração de características em vários níveis. Como cada pirâmide emprega um campo receptivo diferente (filtros diferentes), a fusão de informações globais e sub-regionais é facilitada. Ao contrário, o PANet permite a concatenação de mapas de características próximos do fluxo de *backbone downstream* e do fluxo de característica top-down antes de entrar nos blocos de previsão (*Detection Head*). A rede identifica as coordenadas da *bounding box* junto com a pontuação de confiança para cada classe na detecção do Yolov4 em três escalas diferentes [BOCHKOVSKIY; WANG; LIAO, 2020].

*Bag of Freebies* (BoF) e *Bag of Specials* (BoS) são dois pacotes integrados ao Yolov4 que facilitam o treinamento do modelo para maior desempenho. *Complete Interaction over Union* (CIoU) e *Distance Intersection over Union* (DIoU) são duas das abordagens desses pacotes. Para acelerar a convergência do modelo e aumentar a precisão da regressão da *bounding box*, é aplicado o CIoU. Os recursos geométricos dos *boxes*, incluindo área de sobreposição, proporções e separação entre os pontos centrais do poço, são integrados pelo

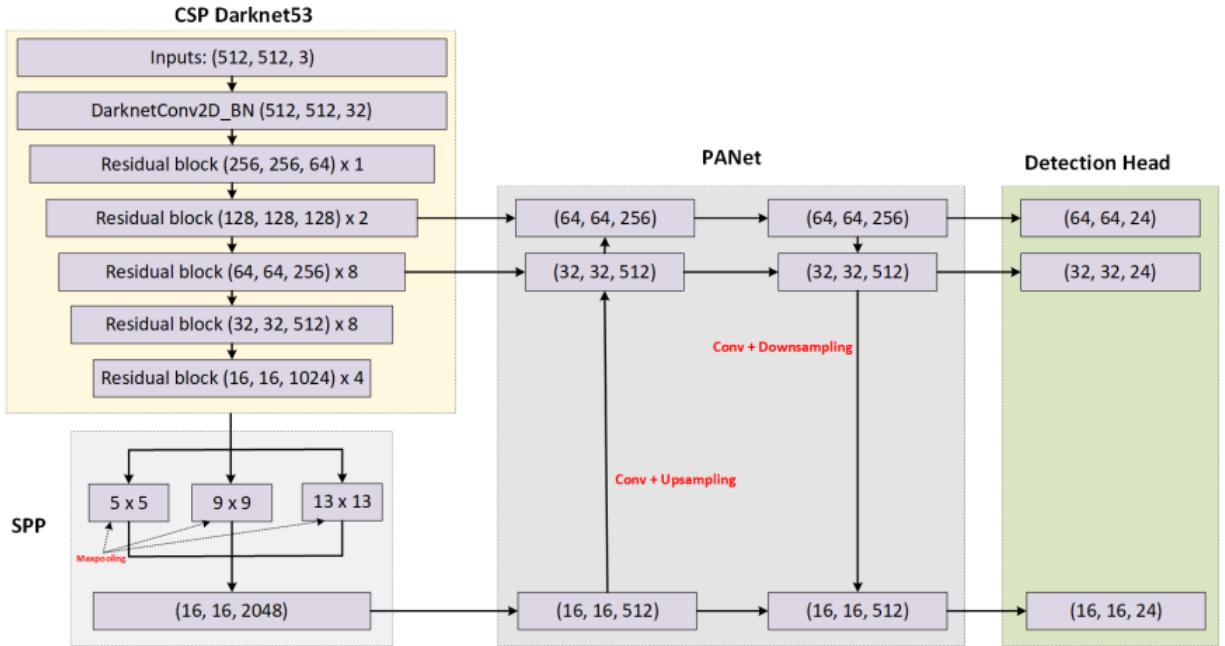


Figura 16 – Visão geral da arquitetura YOLOv4. Formado por quatro blocos, CSP Darknet53, SPP, PANet e *Detection Head*.

Fonte: Erazo et al. [2021].

CIoU [BOCHKOVSKIY; WANG; LIAO, 2020].

Em contraste, DIoU é empregado como uma abordagem NMS, que leva em conta a menor distância entre os centros de duas *bounding boxes* e suprime outros *boxes* redundantes, fortalecendo a resistência do modelo a oclusões [BOCHKOVSKIY; WANG; LIAO, 2020].

O YOLOv4 é um dos detectores que teve bom desempenho no conjunto de dados COCO usando este conjunto de técnicas e metodologias. Em particular, ele levou a um mAP@[.5,.95]=43,5% com uma velocidade de processamento de 65 FPS [BOCHKOVSKIY; WANG; LIAO, 2020].

#### 2.5.4 Yolov4-tiny

A fim de aumentar a velocidade de detecção de objetos, foi desenvolvida a abordagem Yolov4-tiny. Usando uma GPU 1080Ti, o Yolov4-tiny pode detectar objetos a uma taxa de 371 FPS com uma precisão que pode lidar com as demandas de aplicações práticas [JIANG et al., 2020].

No lugar da rede CSPDarknet53 utilizada no método Yolov4, a abordagem Yolov4-tiny usa a rede CSPDarknet53-tiny. O módulo ResBlock na rede residual é substituído pelo módulo CSPBlock na rede CSPDarknet53-tiny. A borda residual de estágio cruzado é usada pelo módulo CSPBlock para integrar as duas partes do mapa de características. Isso aumenta a diferença de correlação das informações de gradiente, permitindo que o

fluxo de gradiente se espalhe ao longo de dois canais de rede distintos. Em comparação com o módulo ResBlock, o módulo CSPBlock pode melhorar a capacidade de aprendizado da rede de convolução. Embora aumente a computação em 10% a 20%, o componente adicionado aumenta a precisão [JIANG et al., 2020].

O diagrama de estrutura da rede YOLOv4-Tiny é mostrado na Figura 17. A rede de *backbone* YOLOv4-Tiny é significativamente mais direta que YOLOv4. Para obter dois tipos de tamanhos de mapas de características para detecção, a *feature pyramid network* é reduzida em 32 vezes e 16 vezes, respectivamente, o que acelera a detecção. A estrutura de rede YOLOv4-Tiny é composta por vários elementos fundamentais. Convolução Conv, normalização BN e função de ativação Leaky-Relu compõem o módulo CBL [WANG et al., 2021].

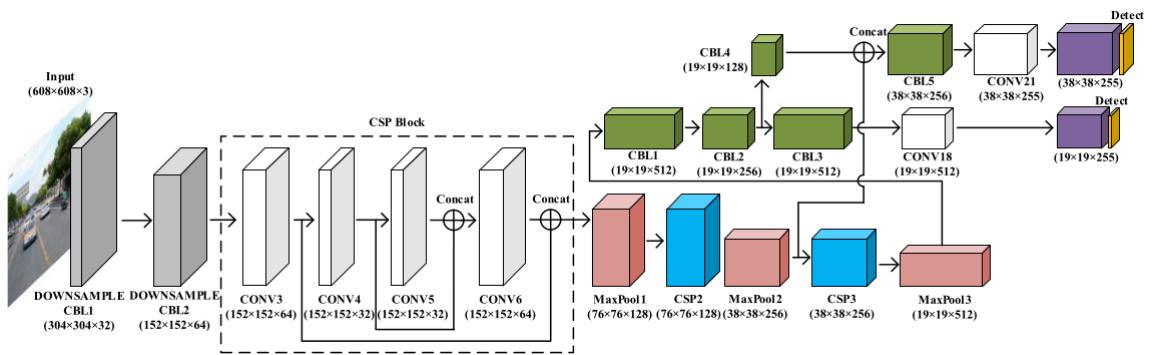


Figura 17 – Estrutura da rede YOLOv4-Tiny.

Fonte: Wang et al. [2021].

Um critério de confiança é utilizado para diminuir o uso redundante de *bounding boxes* no processo de previsão. Se a pontuação de confiança da *bounding boxes* exceder o nível de confiança, ela será mantida; caso contrário, será destruída [JIANG et al., 2020].

## 2.6 Trabalhos Relacionados

A proposta de trabalho de Tomè et al. [2016] propõe um sistema de detecção de pedestres baseado em aprendizado profundo, adaptando uma rede convolucional de uso geral à tarefa em questão. O trabalho traz otimizações na maioria das etapas da detecção de pedestres, superando os métodos tradicionais baseados em atributos projetados a mão (*handcrafted*) e abordagens de aprendizado profundo. A abordagem foi validada implementando-a em um NVIDIA Jetson TK1.

O trabalho apresentado por Cordeiro et al. [2019] propõe um aplicativo para contagem de pessoas por meio da detecção e rastreamento, executando a fase de detecção a cada N quadros, possibilitando rastrear o objeto até alcançar o N-ésimo quadro. Para o

desenvolvimento, os autores utilizaram o Yolov3 e o algoritmo de rastreamento por filtros de correlação. Os resultados foram considerados satisfatórios.

O trabalho de Vargas, Paes e Vasconcelos [2016] tem o objetivo de comparar duas abordagens clássicas para detecção de pedestres relacionadas a Visão Computacional e Rede Neural Convolucional. As técnicas que foram comparadas são: *Features de Haar* com AdaBoost, HOG com SVM e a rede Alexnet. Nos testes realizados, a abordagem com a CNN obteve os melhores resultados na maioria dos casos, comprovando o fato de ser considerada a técnica em estado da arte.

O trabalho de Ize [2019] implementa um aplicativo de rastreamento de múltiplos pedestres usando a abordagem de rastreamento por detecção. Utiliza-se de um modelo pré-treinado para detecção de objetos e reidentificação de pedestres, predição linear de estado com filtragem recursiva, características geométricas e profundas como funções de dissimilaridade entre pedestres, e um algoritmo ótimo para o problema de associação devido à presença de vários pedestres num único quadro de vídeo. O método alcança 79,5% na métrica MOTA.

Outro trabalho relacionado encontrado na literatura é o de [SONG et al., 2018] que propõe um sistema de detecção, rastreamento e reconhecimento de ação de pedestres/carros usando aprendizado profundo usando fluxos de vídeo. Os autores utilizaram a CNN SSD e *mobilenets* para um processo mais rápido e maior taxa de detecção.

# 3 Materiais e Métodos

Este capítulo apresenta a estrutura geral do desenvolvimento deste projeto. A Seção 3.1 retrata sobre o *dataset* adquirido para a detecção e rastreamento dos pedestres. Na Seção 3.2 são exibidos algoritmos, *frameworks*, bibliotecas e plataformas online utilizados para o andamento do projeto. E na Seção 3.3 se aborda a maneira como o trabalho foi desenvolvido.

## 3.1 Dataset

Para o presente trabalho utilizou-se uma câmera pública disponibilizada no site da Catve (Cascavel TV Educativa) [CATVE, 2022]. Essa câmera registra em tempo real a Aduana da Ponte Internacional da Amizade. A transmissão ao vivo é feita por meio de um arquivo que tem a extensão “.m3u8” utilizando uma URL, gerado pelo servidor HTTP (*Hypertext Transfer Protocol*) . Essa URL é adicionada no Fluxo de Rede do VLC [VIDEOLAN, 2022] e dessa forma é possível ter acesso à transmissão em tempo real pelo próprio computador.

Com base nisso, foram gravados vídeos entre 5 a 30 minutos utilizando o OBS Studio [JIM, 2022] durante 12 dias, entre as 8:00 horas da manhã e 16:00 horas da tarde, onde há um fluxo maior de pedestres. Os vídeos foram gravados em diversas condições de iluminação e clima, como tempo ensolarado, chuvoso, nublado ou parcialmente ensolarado. A Figura 18, ilustra a Aduana em diferentes tempos climáticos. Ao todo foram 21 GB de vídeos registrados para o *dataset*, da qual foram extraídas 1600 imagens por meio de um *script* em Python. Importante ressaltar que o conjunto de teste foi construído em 3 dias diferentes do conjunto de treinamento.

Como a câmera não tem uma qualidade ou resolução avançada, muitas imagens foram excluídas já que haviam distorções no formato dos pedestres atravessando a via. A Figura 19 apresenta uma imagem distorcida contendo um pedestre atravessando a via. As imagens que apresentaram essas distorções não foram inseridas no *dataset*, pois poderiam dificultar o aprendizado da CNN. Inicialmente, as imagens tinham o tamanho 1800 x 905 pixels.



Figura 18 – Tempos climáticos diferentes registrados.

Fonte: A própria autora.



Figura 19 – Distorção na imagem.

Fonte: A própria autora.

## 3.2 Ferramentas

### 3.2.1 Pré-processamento

Para aplicar técnicas de pré-processamento, o Roboflow [ROBOFLOW, 2022] foi escolhido por ser uma plataforma para visão computacional que oferece melhores métodos para coleta de dados, pré-processamento e treinamento de modelos, permitindo que os usuários criem modelos de visão computacional com mais rapidez e precisão. Os usuários do Roboflow podem fazer *upload* de seus próprios conjuntos de dados, anotar, alterar as orientações da imagem, redimensionar imagens, alterar o contraste da imagem e enriquecer os dados.

O Roboflow também inclui uma ferramenta universal de conversão de anotações que permite aos usuários fazer *upload* e converter anotações de um formato para outro

sem a necessidade de criar *scripts* de conversão para conjuntos de dados de detecção de objetos exclusivos.

### 3.2.2 LabelImg

O software escolhido para a realização da tarefa de anotação de objetos em imagens foi o LabelImg [TZUTALIN, 2022], disponível sob licença gratuita em seu repositório web. Dentre as funcionalidades existentes, permite a demarcação de *bounding boxes* nos objetos de interesse em cada imagem, as anotações são gravadas no formato PASCAL VOC compatível com ImageNet como arquivos XML e também, suporta os formatos CreateML e Yolo. É uma maneira simples e gratuita de anotar o conjunto de imagens.

Na Figura 20, é possível observar a interface do sistema. À esquerda, está as opções variadas que se podem aplicar nas imagens, como “*Verify Image*”, quando não há objetos de interesse na imagem ou “*Create RectBox*”, quando deseja-se demarcar um objeto de interesse. Do lado direito, é possível ver a classe “person”. Ao meio, nota-se quatro *bounding boxes* com a classificação “person” na imagem.



Figura 20 – LabelImg.

Fonte: A própria autora.

### 3.2.3 Treinamento

Para criar o sistema de detecção e classificação, é necessário um algoritmo de rede neural convolucional para realizar a detecção. A CNN Yolov4-tiny e o *framework* Darknet foram selecionados para o presente trabalho, pois a aplicação envolve vídeos e uma transmissão ao vivo. A Yolo tem se destacado no estado da arte, sendo mais rápido

no treinamento e inferência, alcançando bons resultados (mAP) e se propõe a detectar objetos em tempo real.

Darknet é uma rede neural de código aberto escrita em C e CUDA, suporta computação de CPU e GPU. Darknet pode ser usado diretamente e não precisa de nenhuma outra biblioteca. Possui 2 dependências opcionais, *OpenCV* para uma variedade maior de tipos de imagem suportados e CUDA para computação GPU. O Darknet é famoso por sua arquitetura de detecção de objetos em tempo real: Yolo.

O Google Colab foi utilizado para realizar os treinamentos da rede neural. O Colab é baseado no Jupyter Notebook para realizar operações de aprendizado de máquina e aprendizado profundo. O Google Colab fornece uma GPU de acesso gratuito em tempo de execução. A GPU disponível na plataforma Colab é o modelo Tesla K80, P100, P4 e T4 com capacidade computacional de 7,5 GHz com 14 GB de RAM.

### 3.2.4 Rastreamento

Para o processo de rastreamento e contagem de pedestres foi utilizado um notebook contendo uma GPU da Nvidia GeForce GTX 1060 Mobile, bibliotecas como CUDA, cuDNN e *OpenCV*, como também a linguagem de programação Python. A NVIDIA criou uma biblioteca CUDA [CORPORATION, 2022] para aceleração de GPU voltada para redes neurais profundas. A cuDNN (*NVIDIA CUDA Deep Neural Network*) pode ser utilizada para aprimorar implementações de aprendizagem profunda como convoluções, *pooling*, normalização e camadas de ativação.

O *OpenCV* (*Open Source Computer Vision Library*) é uma biblioteca de software de visão computacional e aprendizado de máquina de código aberto com interfaces C, C++, Python e Java. Ele implementa um grande conjunto de algoritmos de visão computacional de última geração, incluindo o módulo DNN (*Deep Neural Network*). Este é o módulo do *OpenCV* que está relacionado ao aprendizado profundo. Este módulo permite o uso de modelos pré-treinados para inferência de *frameworks* como Caffe, TensorFlow, Torch e Darknet. Isso significa que é possível treinar modelos usando o Darknet e fazer inferência/previsão apenas com o *OpenCV*, tornando o código final muito mais compacto e simples. Pode também fornecer resultados de inferência mais rápidos para a CPU. A Figura 21 apresenta a função que lê o modelo de rede armazenado no *framework* Darknet. Além de suportar GPU da NVIDIA baseada em CUDA, o módulo DNN do *OpenCV* também suporta GPUs Intel baseadas em OpenCL.

```
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
```

Figura 21 – Função do módulo DNN.

Fonte: A própria autora.

O algoritmo SORT (*Simple Online RealTime Tracking*) [BEWLEY et al., 2016] foi selecionado, pois funciona com o princípio de rastreamento por detecção. O SORT utiliza um vetor de *bounding boxes* como entrada e consegue rastrear estes múltiplos objetos em um vídeo em tempo real, alcançando bons resultados com baixo processamento, devido a utilização de métodos mais simples, porém ainda eficientes. O SORT é composto por quatro componentes principais, sendo eles: detecção, estimativa, associação e manutenção da vida útil de objetos rastreados.

- Detecção: a eficácia da detecção afeta significativamente o desempenho de um rastreamento. De acordo com Bewley et al. [2016], que testaram vários detectores usando dois rastreadores, os melhores resultados de rastreamento foram alcançados quando o detector mais preciso foi usado. Os elementos encarregados de classificar e segmentar as coisas são os detectores. As redes neurais convolucionais são normalmente empregadas como detectores. Os detectores Fast R-CNN e Yolo são dois exemplos [ANGHINONI, 2020].
- Estimativa: Este elemento descreve o modelo do objeto, ou o modelo de movimento que é utilizado para levar a identificação do objeto aos quadros seguintes. Um filtro de Kalman é usado para estimar a velocidade linear e os dados de posição no modelo [BEWLEY et al., 2016; ANGHINONI, 2020].
- Associação: Para associar detecções a objetos existentes, cada *bounding box* é estimada pela sua geometria do *frame* anterior, predizendo a localização no *frame* atual. A IoU entre cada detecção real e a *bounding box* antecipada deve então ser calculada. O Algoritmo Húngaro é usado para realizar a associação [BEWLEY et al., 2016].
- Manutenção de objetos: Identificadores devem ser criados e destruídos conforme os objetos entram e saem da cena. Qualquer detecção com uma IoU acima do valor mínimo exigido é levada em consideração ao criar identificadores. Alguns são usados para observar essa detecção e, uma vez que um item tenha um modelo de objeto claro e alguns relacionamentos consecutivos, ele é considerado um novo objeto. Os identificadores são removidos se estiverem ausentes por vários *frames* seguidos [BEWLEY et al., 2016].

O SORT funciona bem em termos de precisão, apesar da eficácia do Filtro Kalman, a abordagem retorna muitas alterações de identidade e tem um desempenho ruim em cenários com oclusão [ANGHINONI, 2020].

### 3.3 Métodos

A seleção e rotulagem das imagens são uma das etapas mais caras em termos de tempo de dedicação, já que um bom número de imagens deve ser selecionado manualmente. Com a plataforma Roboflow, as imagens foram redimensionadas de 1800 x 905 para 832 x 832 pixels, com o foco de manter a padronização durante o treinamento. Para evitar erros com a orientação da imagem, foi aplicado a Auto Orientação e para aumentar o número de exemplos, aplicou-se o *Bounding box flip horizontal*, basicamente inverte ou espelha somente o objeto de interesse. Dessa forma, fornece-se exemplos extras de pedestres atravessando a via em direções diferentes.

Para o treinamento é importante alterar alguns hiperparâmetros buscando melhorar a detecção de objetos. Hiperparâmetros são parâmetros de modelos que devem ser definidos antes de treinar o modelo. Os hiperparâmetros ficam armazenados em um arquivo “.cfg”. Existem diversos tipos de arquivos cfg para diferentes objetivos. Observou-se durante a fase de anotação de imagens, que havia diferentes tamanhos do objeto de interesse, os pedestres podiam estar mais visíveis (perto da câmera) ou mais distantes. Bochkovskiy [2022], um dos criadores da Yolo e Darknet, disponibiliza um arquivo cfg, com hiperparâmetros específicos para yolov4 e yolov4-tiny que trata especificamente desse caso para melhorar a detecção de objetos, adicionando mais uma camada Yolo. A Tabela 1 apresenta as diferenças entre os arquivos cfg yolov4-tiny padrão e yolov4-tiny específico. Vale ressaltar que alguns parâmetros são diferentes por consequência do aumento do número de camadas.

Tabela 1 – Diferenças entre arquivos Yolov4-tiny.

	Yolov4-tiny padrão	Yolov4-tiny específico
Camada convolucional	21	24
Camada yolo	2	3
Route	11	13
Upsample	1	2

Fonte: A própria autora.

Alguns hiperparâmetros foram alterados de acordo com a necessidade atual. Os hiperparâmetros alterados são: *Batch*, *Subdivisions*, *width* e *height*, *Max\_batches*, *steps* e *Random*. Os valores setados para esses hiperparâmetros são valores recomendados pelo Bochkovskiy [2022]. Na tabela Tabela 2 é apresentado os valores utilizados em todos os treinamentos.

Bochkovskiy [2022] traz várias recomendações para melhorar a detecção de objetos, as que foram inseridas nesse trabalho são:

- *Random* = 1, aumentará a precisão treinando a Yolo para diferentes resoluções.

Tabela 2 – Hiperparâmetros alterados nos treinamentos.

hiperparâmetros	Valores utilizados
<i>Batch</i>	64
<i>Subdivisions</i>	16
<i>width e height</i>	416, 512, 608, 812
<i>Max_batches</i>	6000
<i>steps</i>	4800, 5400
<i>Random</i>	1

Fonte: A própria autora.

- Aumentar a resolução de rede,  $width = 608$  e  $height = 608$  ou qualquer valor múltiplo de 32 aumentará a precisão.
- Checar se cada objeto que deseja detectar é rotulado obrigatoriamente em seu conjunto de dados.
- Para cada objeto que se deseja detectar deve haver pelo menos 1 objeto semelhante no conjunto de dados de treinamento com aproximadamente o mesmo: formato, lado do objeto, tamanho relativo, ângulo de rotação, inclinação, iluminação.
- Importante que o conjunto de dados de treinamento inclua imagens com objetos não rotulados que não se deseja detectar, amostras negativas sem *bounding box*.

Uma das maneiras de evitar o *overfitting* é utilizar o método de Validação Cruzada K-fold, onde usa parte dos dados disponíveis para ajustar o modelo e uma parte diferente para testá-lo. Divide-se os dados em K partes de tamanhos aproximadamente iguais. A Figura 22 ilustra o processo que é realizado.

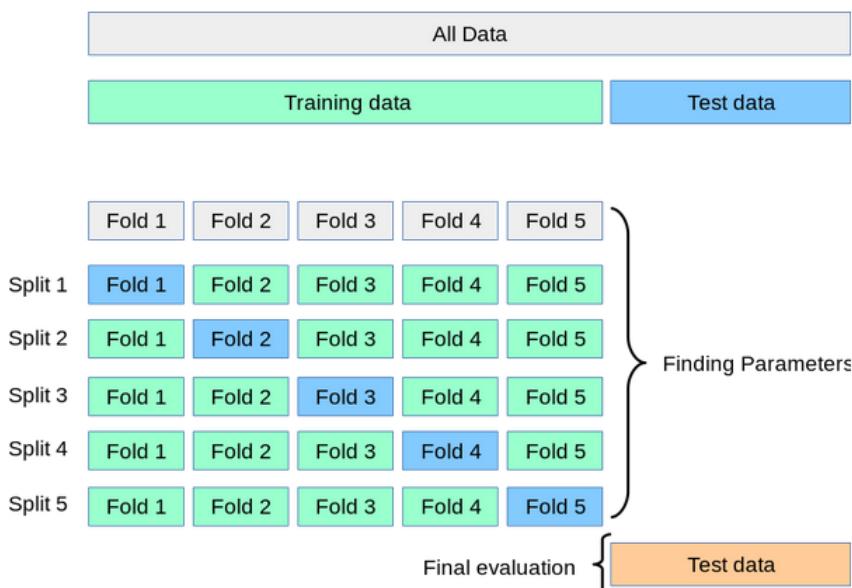


Figura 22 – Validação Cruzada K-fold.

Fonte: Developers [2022].

A Figura 23 apresenta um fluxograma com o objetivo de esclarecer todo o processo realizado no presente trabalho. A fase 1 condiz com a escolha da câmera e elaboração do conjunto de imagens a ser utilizado. A fase 2 está relacionada com as anotações dos objetos de interesse e aplicação de pré-processamento. A fase 3 foi o momento de realizar diversos treinamentos para melhor compreender os dados, a utilização do método de Validação Cruzada e analisar os resultados. Por fim, o modelo final é utilizado na fase 4 para rastreamento e contagem de pedestres realizando travessias de risco.

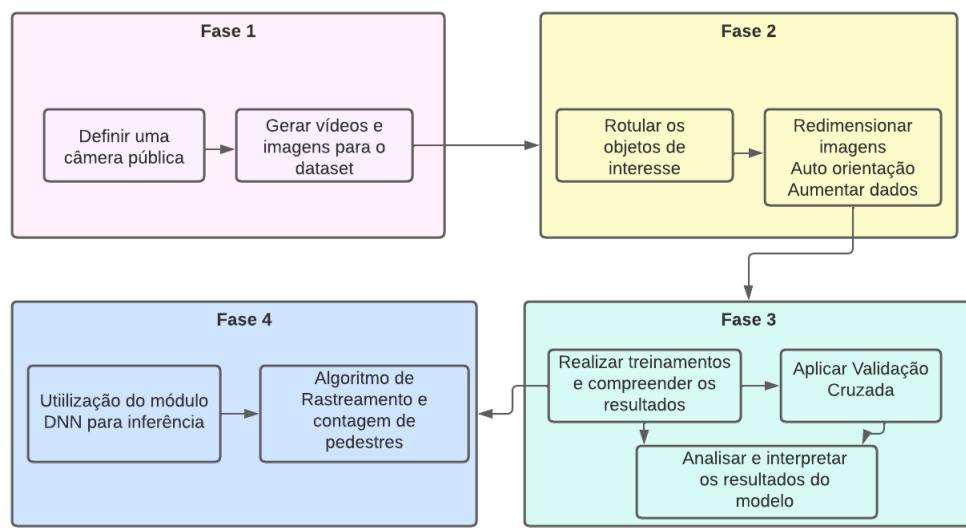


Figura 23 – Metodologia utilizada no desenvolvimento do trabalho.

Fonte: A própria autora.

# 4 Resultados e Discussões

Neste capítulo são apresentados os resultados obtidos avaliados com diversas métricas. A seção 4.1 trás definições das métricas utilizadas para avaliar o modelo e a seção 4.2 apresenta o processo de treinamento realizado para o aprendizado da CNN e exibe os resultados do modelo final.

## 4.1 Métricas de Avaliação

Um algoritmo de AM supervisionado geralmente é avaliado observando o desempenho de seu preditor ao categorizar novos itens que não foram apresentados anteriormente durante o treinamento [FACELI et al., 2010]. As previsões Verdadeiro positivo (VP) , Falso positivo (FP) e Falso negativo (FN) de um classificador são relatadas na matriz de confusão, que é apenas uma matriz quadrada, em que:

- VP significa a quantidade de verdadeiros positivos, ou instâncias da classe positiva que foram identificadas com precisão.
- FP significa falsos positivos, ou o número de instâncias cuja verdadeira classe é negativa, mas que foram erroneamente atribuídas à classe positiva.
- FN representa o número de falsos negativos, ou o número de casos que foram inicialmente previstos como sendo da classe positiva, mas acabaram sendo da classe negativa.

Várias medidas de desempenho diferentes podem ser obtidas a partir da matriz de confusão. Estas são algumas delas:

- A **taxa de erro de classe positiva**, também conhecida como taxa de falso negativo, é a porcentagem de exemplos de classe positiva que o preditor classificou erroneamente [FACELI et al., 2010].

$$TFN = \frac{FN}{VP + FN} \quad (4.1)$$

- A **taxa de erro total** é calculada dividindo o valor da diagonal secundária da matriz pelo valor de toda a matriz [FACELI et al., 2010].

$$err = \frac{FP + FN}{n} \quad (4.2)$$

- A **precisão** é a porcentagem de casos positivos entre todos aqueles que se prevê serem positivos e que são realmente identificados como tal [FACELI et al., 2010].

$$precisão = \frac{VP}{VP + FP} \quad (4.3)$$

- A taxa de acerto na classe positiva corresponde à **sensibilidade** ou **revocação**. Também conhecido como a taxa de verdadeiro positivo.

$$revocação = \frac{VP}{VP + FN} \quad (4.4)$$

Sensibilidade e precisão podem ser vistos como indicadores da completude e exatidão do modelo, respectivamente. Como a precisão e a sensibilidade são normalmente abordadas em conjunto e não separadamente, uma medida como a medida  $F_1$ , que é a média harmônica ponderada da precisão, é usada [FACELI et al., 2010].

$$F_1 = \frac{2 * (precisão * revocação)}{precisão + revocação} \quad (4.5)$$

A métrica IoU é usada na comparação das posições detectadas do modelo com as posições reais dos objetos, sendo a medida de avaliação mais comum usada na segmentação, detecção de objetos e rastreamento, também conhecido como índice Jaccard. O IoU é usado para avaliar a precisão de um detector de objetos ou rastreador em um conjunto de dados. Em geral, ele pode ser usado para medir a saída de qualquer algoritmo que retorne resultados como *bounding boxes*. Para contabilizar a sobreposição dessas regiões, seu cálculo leva em consideração as áreas esperadas e reais [MARQUES, 2019].

A interseção entre a área da região do objeto e a área da previsão deve ser dividida pela união para produzir o IoU, conforme mostrado na Figura 24. Com valores percentuais, o algoritmo pode ser limitado por um IoU maior, sendo a seleção de objetos é mais precisa. Por outro lado, um valor baixo faz com que o algoritmo tenha uma margem de aceitação alta, mas uma precisão menor [MARQUES, 2019].

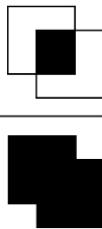
$$\text{IoU} = \frac{\text{Área da interseção}}{\text{Área da união}}$$


Figura 24 – Exemplificando o cálculo da métrica IoU.

Fonte: [MARQUES, 2019]

Cálculos de precisão e sensibilidade para a tarefa de identificação de objetos devem levar em consideração um limiar (*threshold*) de IoU específico. Em outras palavras, o IoU

entre uma previsão e um objeto deve ser maior que o limiar selecionado para que seja considerado um Verdadeiro Positivo real. Geralmente, os limiares podem ser 50%, 75% e 95%. Para o presente trabalho, o limiar utilizado foi de 50% [MARQUES, 2019].

Calcular a precisão e a revocação em cada limite de confiança permite preencher uma curva de Precisão-Revocação (PR) . A precisão média ou *Average Precision* (AP) é a área sob esta curva. A curva PR é condensada por AP em um valor escalar. Quando a revocação e a precisão são altas, a precisão média também é alta, e quando qualquer uma delas é baixa dentro de um intervalo de valores de limite de confiança, ela é baixa. Entre 0 e 1 é o intervalo AP. O mAP é a métrica mais utilizada em detecção de objetos, é a média da precisão média de cada classe. Como no presente trabalho só há uma classe, AP é o termo utilizado [MARQUES, 2019].

## 4.2 Resultados

Inicialmente, realizou-se um treinamento utilizando 376 imagens para compreender as limitações da CNN. Observou-se que a CNN detectava e classificava ciclistas e motociclistas como pedestres. Com isso, foram adicionadas 396 exemplos negativos que continham principalmente motociclistas.

No *dataset* final obteve-se 1600 imagens, na qual foi dividido em 90% para treino e validação e 10% para teste. O *dataset* correspondente aos 90% foram divididos em 5 partes ou *folds*, na qual o conjunto de validação nunca é o mesmo, sendo 1152 exemplos para treinamento, 288 para validação e 160 para teste, garantindo que cada pedestre no conjunto de dados esteja em um conjunto de validação. Para cada *fold* foram aplicados *Bounding box flip horizontal* para aumentar o número de exemplos. Para o modelo final, o conjunto de treinamento e validação foram reunidos resultando em 1440 imagens para treino. A técnica de *transfer learning* foi utilizada em todos os treinamentos realizados.

A Tabela 3 apresenta os resultados de AP, IoU e F1-Score de cada *fold*. Em AM, é comum utilizar a técnica de *Grid Search* para encontrar a combinação dos melhores parâmetros, no entanto, o processo seria muito custoso para este trabalho. O único parâmetro que foi alterado em cada *fold* é a resolução da rede (*width* e *height*), pois de acordo com o Bochkovskiy [2022] pode aumentar a precisão. No caso não é possível validar isso, pois teria que realizar treinamentos com diferentes resoluções com o mesmo *fold* para verificar se de fato aumenta da precisão. As resoluções da rede mais utilizadas na literatura são 416, 512, 608 e 832. É possível observar que a métrica IoU tem um melhor resultado conforme aumenta a resolução da rede. A Tabela 4 apresenta o desempenho médio estimado e o desvio padrão de cada métrica.

Para o último treinamento utilizando o conjunto de teste, optou-se pelo valor 832

Tabela 3 – Resultados dos 5 folds.

Folds	width x height	AP50	IoU	F1-Score
Fold 1	416 x 416	81.78%	49.02%	0.74
Fold 2	512 x 512	69.67%	52.67%	0.70
Fold 3	608 x 608	77.17%	62.67%	0.75
Fold 4	832 x 832	71.32%	57.76%	0.71
Fold 5	832 x 832	80.22%	65.28%	0.76

Fonte: Autor

Tabela 4 – Resultados da média e desvio padrão para as métricas AP, IoU e F1-Score.

	Média	Desvio Padrão
AP	76.032	4.8
IoU	60.468	6.7
F1-Score	73.2	2.3

Fonte: Autor

no hiperparâmetro *width* e *height*, pois teve o melhor resultado nas métricas IoU e F1-Score e o segundo melhor AP. A vantagem de ter um conjunto de teste que o modelo não tenha visto antes, durante as etapas de treinamento é que pode-se obter uma estimativa menos tendenciosa de sua capacidade de generalizar para novos dados. O conjunto de treinamento recebeu mais 76 exemplos utilizando a técnica para aumentar os dados, resultando em 1516 imagens.

O modelo mostrou-se eficiente em detectar pedestres realizando travessias em diversos ambientes climáticos. Demonstrando a capacidade de localizar objetos mesmo quando ocorre oclusão parcial. Os resultados obtidos durante os testes de classificação das imagens mostraram-se promissores.

Dos 303 pedestres identificados atravessando a via no conjunto de teste, 274 foram corretamente classificados (VP). O que não foi classificado corretamente: 29 não foram detectados pela rede (FN) e 138 foram identificados como pedestres atravessando a via, mas não era (FP). Os valores obtidos foram 90% de sensibilidade, 67% de precisão, 9.6% da taxa de erro da classe positiva e 38% da taxa de erro total.

O número alto de Falso Positivo não é porque a CNN está detectando motociclistas ou ciclistas, mas se deve pelo fato de várias pessoas estarem no encostamento da via ou no meio dela, com o objetivo de entregar panfletos. Essa prática é muito comum na Aduana, pois muitas pessoas vão ao Paraguai para fazer compras.

A Figura 25 e a Figura 26 mostram várias pessoas no meio da via. Na Figura 25, a CNN detecta dois pedestres realizando uma travessia de risco, no entanto a terceira detecção com confiança 0.43, não é considerado para o trabalho. É possível observar que há vários motociclistas na foto e a CNN não detecta nenhum. Na Figura 26, a CNN detecta 7 pessoas que estão no meio da via, mas que não é considerado para este trabalho

e realiza 2 detecções VP.



Figura 25 – Imagem do conjunto de teste que ilustra diversas pessoas na via.

Fonte: A própria autora.



Figura 26 – Imagem do conjunto de teste que ilustra diversas pessoas na via sendo detectadas pela CNN.

Fonte: A própria autora.

Entre as 160 imagens de teste, a CNN não detectou nenhum motociclista como pedestre, mas detectou erroneamente 1 ciclista como pedestre, apresentado na Figura 27. Isso se deve pelo fato que a maioria dos exemplos negativos são de motociclistas, em que há maior frequência na via.

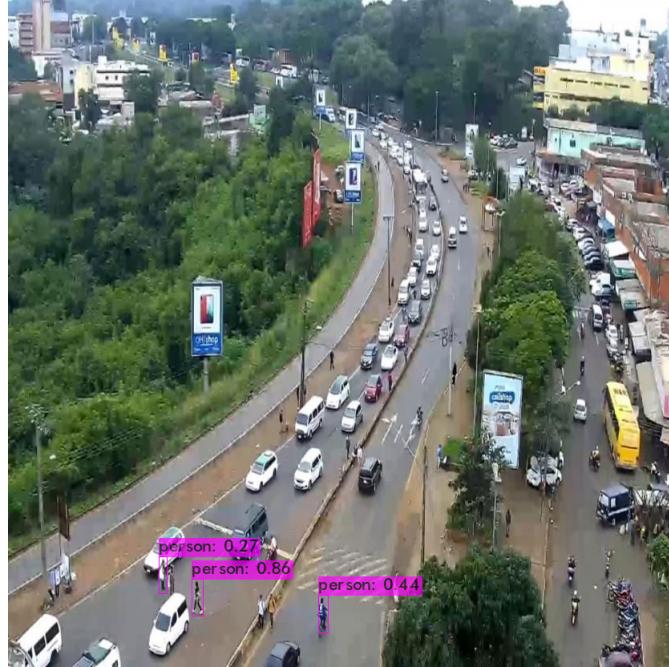


Figura 27 – Ciclista detectado pela CNN.

Fonte: A própria autora.

Outro FP encontrado foi a detecção sombras como pedestre em 9 imagens. A Figura 28 ilustra esse FP. Essa sombra aparece em quase todas as imagens com o dia ensolarado no conjunto de teste.

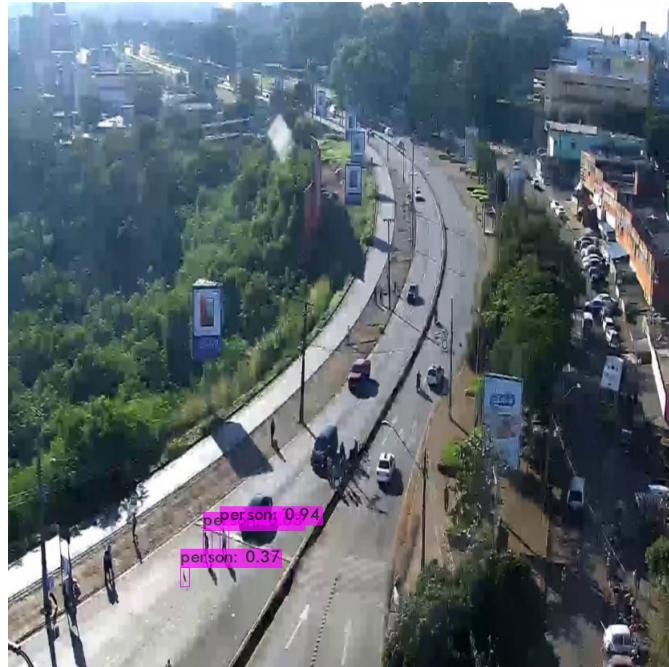


Figura 28 – Estranho FP que a CNN detectou.

Fonte: A própria autora.

A Tabela 5 apresenta os resultados das métricas IoU, AP, F1-Score e outras já

citadas. É possível compreender que a precisão resultou em 67% pelo número considerável de FP e consequentemente interferiu no resultado de F1-Score e AP. Ainda sim, obteve-se quase 89% de precisão média da CNN, considerado um resultado positivo. A Figura 29 apresenta o gráfico com o AP e a função de perda (*loss*) ao longo das 6000 iterações.

Tabela 5 – Resultados do modelo final.

Modelo final	
AP50	88.76%
IoU	50.86%
F1-Score	77%
Precisão	67%
Recall	90%

Fonte: A própria autora.

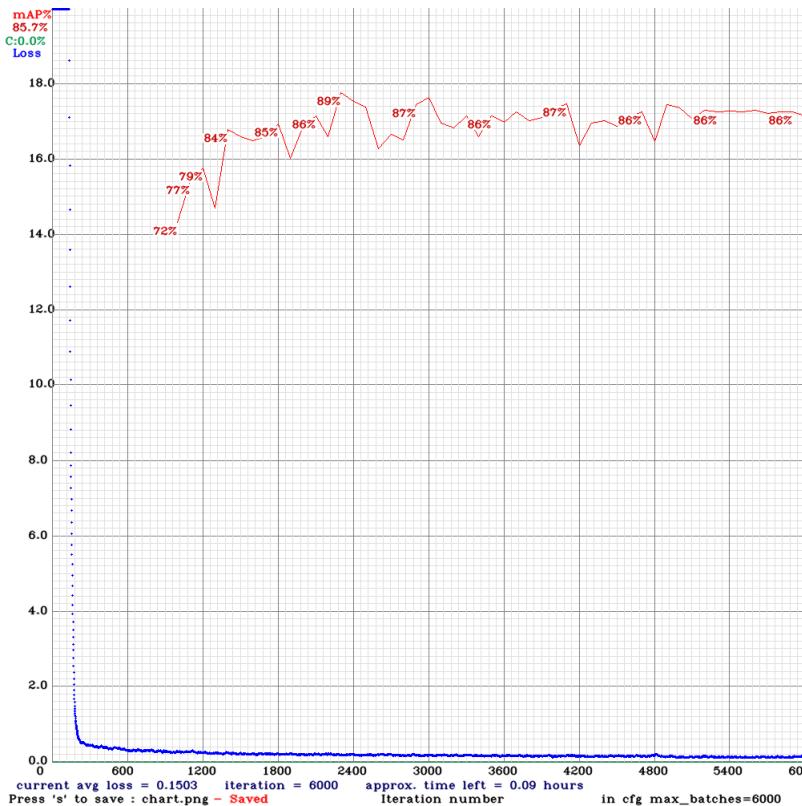


Figura 29 – Gráfico que relaciona AP com a função de perda.

Fonte: A própria autora.

O algoritmo Sort para rastreamento de objetos é utilizado para auxiliar na contagem de pedestres realizando uma travessia de risco. O parâmetro que define o número máximo de *frames* que mantém os identificadores de um objeto sem detecções associadas foi alterado para 50, visto a necessidade da aplicação. O valor mínimo do IoU definido é 0.5, somente detecções com o IoU acima de 0.5 foram consideradas. Para realizar a contagem foram traçadas 3 linhas no meio da rodovia utilizando o OpenCV. A Figura 30 ilustra as linhas projetadas na imagem. Ao passar um pedestre por uma dessas linhas é



Figura 30 – Linhas amarelas traçadas para contagem de pedestres.

Fonte: A própria autora.

realizado a contagem. O tempo de execução da aplicação varia de 11 a 13 FPS. O ideal para a aplicação em tempo real seria um valor mínimo próximo de 30 FPS. Uma maneira de solucionar esse problema seria executar a aplicação em uma GPU mais avançada. No entanto, essa possibilidade nem sempre é possível.

Ao executar a aplicação foi possível observar que a CNN detecta motociclistas com pouca frequência. Quando uma contagem é feita, o *frame* correspondente é salvo como imagem e a contagem é registrada em um arquivo csv. A Figura 31 e a Figura 32 apresentam a contagem de pedestres atravessando a via de transporte. A Figura 32 ilustra um exemplo de pedestre atravessando na faixa e sendo detectado pela CNN, no entanto a contagem não é feita. A Figura 33 apresenta pedestre atravessando a via, mas a contagem não é realizada. Alguns fatores que podem dificultar na contagem são: a qualidade da câmera e a CNN não detectar o pedestre ao passar pela linha amarela.



Figura 31 – Contagem de pedestre realizando travessia de risco.

Fonte: A própria autora.



Figura 32 – Contagem de pedestre realizando travessia fora da faixa de pedestre.

Fonte: A própria autora.

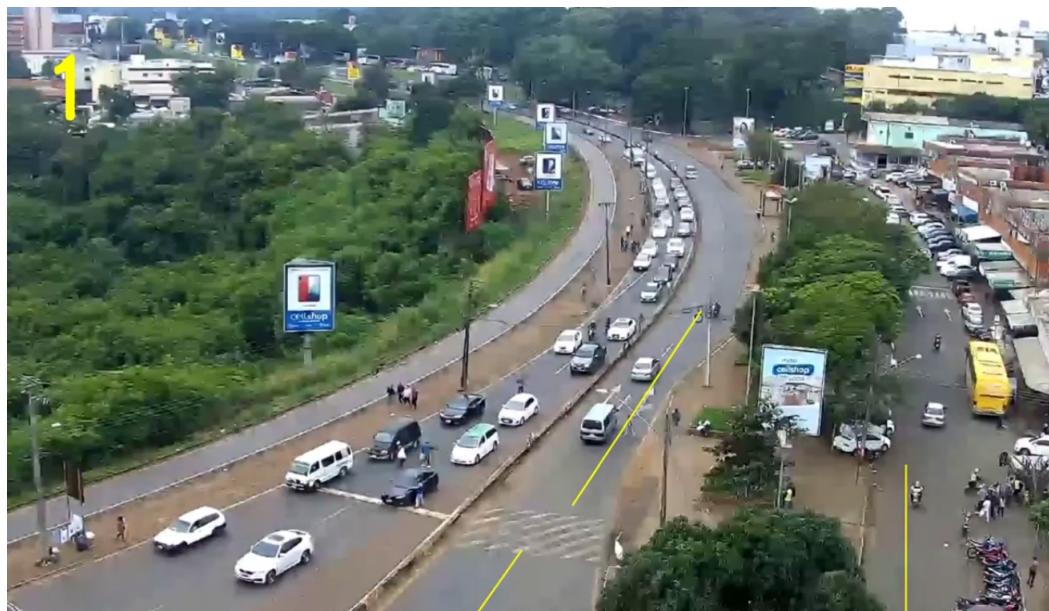


Figura 33 – Pedestre atravessando fora da faixa, mas a contagem não é feita.

Fonte: A própria autora.

## 5 Conclusão

Neste trabalho, foi utilizado a CNN Yolo para detectar pedestres realizando travessias de risco, como também realizar a contagem desses pedestres por meio do rastreamento de objetos em um cenário real, em tempo real. A ideia principal é gerar informações de valor que possam ser apresentadas as autoridades devidas e que medidas de segurança possam ser tomadas e uma conscientização maior na população sobre o trânsito.

Para realizar a detecção de pedestres é indispensável uma placa de vídeo. Além disso, a tecnologia CUDA é necessária, pois influencia o tempo de treinamento ao paralelizar os cálculos nos núcleos da placa. Embora existam outras ferramentas que podem fazer isso, os *frameworks* mais conhecidos fazem uso do CUDA porque é um recurso bem desenvolvido.

As redes neurais convolucionais são os algoritmos ideais para a tarefa de detecção de objetos, de acordo com os estudos e resultados obtidos. O treinamento do objeto de interesse foi simplificado através da implementação do modelo de rede Yolov4-tiny, utilizando a rede pré-treinada e usufruindo do conhecimento já aprendido pelo modelo.

Com os diversos treinamentos realizados, o modelo final demonstrou resultados satisfatórios, como a média da precisão de 89% aproximadamente, a média do IoU de 50% e a média harmônica ponderada da precisão de 77%. Seguir as instruções dos autores da Yolo de como melhorar a precisão da detecção de objetos contribuiu para esse resultado final. Utilizar a rede da Yolov4-tiny com três camadas Yolo, alterar os parâmetros *random* e *width* e *height*, adicionar exemplos negativos influenciaram positivamente nos resultados.

Contudo, ainda é possível melhorar ainda mais esses resultados, adicionando mais exemplos negativos de ciclistas, motociclistas e pessoas que trabalham entregando panfletos na rodovia, testando diferentes hiperparâmetros. Isso é crucial para o algoritmo de Sort que depende das detecções do modelo para rastrear os objetos.

Como trabalho futuro, essa solução ser testada com diferentes técnicas de rastreamento que podem influenciar numa contagem mais precisa dos pedestres. Outro trabalho que pode ser desenvolvido é aplicação em diferentes regiões da cidade.

# Referências

- AGGARWAL, C. C. *Neural Networks and Deep Learning*. [S.l.]: Springer, 2018. Citado 12 vezes nas páginas 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21 e 23.
- ANGHINONI, L. A. N. *Comparaçao de técnicas de rastreamento de objetos*. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2020. Citado 2 vezes nas páginas 26 e 38.
- BEWLEY, A. et al. Simple online and realtime tracking. In: IEEE. *2016 IEEE international conference on image processing (ICIP)*. [S.l.], 2016. p. 3464–3468. Citado na página 38.
- BLOCK, H. D.; KNIGHT, B.; ROSENBLATT, F. Analysis of a four-layer series-coupled perceptron. *Reviews of Modern Physics*, APS, v. 34, n. 1, p. 135, 1962. Citado na página 5.
- BOCHKOVSKIY, A. *How to improve object detection*. <https://github.com/AlexeyAB/darknethow-to-improve-object-detection>: [s.n.], 2022. Citado 2 vezes nas páginas 39 e 44.
- BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. Citado 2 vezes nas páginas 30 e 31.
- BUCHANAN, B.; SUTHERLAND, G.; FEIGENBAUM, E. Heuristic dendral: A program for generating explanatory hypotheses in organic chemistry. In: . [S.l.]: Edinburgh Univ Press, 1969. p. 209–54. Citado na página 5.
- CATVE. *Câmeras ao vivo*. <https://catve.com/aduana>: [s.n.], 2022. Citado na página 34.
- CHAKRABORTY, A.; MUKHERJEE, D.; MITRA, S. Development of pedestrian crash prediction model for a developing country using artificial neural network. *International journal of injury control and safety promotion*, Taylor & Francis, v. 26, n. 3, p. 283–293, 2019. Citado na página 1.
- CORDEIRO, A. F. et al. Rastreamento e contagem de pedestre em tempo real por meio de imagens digitais. In: SBC. *Anais do XVI Congresso Latino-Americano de Software Livre e Tecnologias Abertas*. [S.l.], 2019. p. 146–149. Citado na página 32.
- CORPORATION, N. *CUDA Toolkit*. <https://developer.nvidia.com/how-to-cuda-c-cpp>: [s.n.], 2022. Citado na página 37.
- CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, Springer, v. 2, n. 4, p. 303–314, 1989. Citado na página 14.
- DAWSON-HOWE, K. *A practical introduction to computer vision with opencv*. [S.l.]: John Wiley & Sons, 2014. Citado na página 25.

- DEVELOPERS, S. learn. *Cross-validation: evaluating estimator performance.* [https://scikit-learn.org/stable/modules/cross\\_validation.html#cross-validation](https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation) : [s.n.], 2022. Citado na página 40.
- DINOV, I. D. *Data Science and Predictive Analytics.* [S.l.]: Springer, 2018. Citado na página 12.
- ERAZO, J. J. M. et al. Desenvolvimento de um sistema de contagem e classificação de veículos utilizando redes neurais convolucionais. 2021. Citado 3 vezes nas páginas 28, 30 e 31.
- FACELI, K. et al. *Inteligência artificial: uma abordagem de aprendizado de máquina.* [S.l.]: LTC, 2010. Citado 11 vezes nas páginas 6, 7, 8, 9, 10, 14, 15, 16, 17, 42 e 43.
- FERNANDES, C. M.; BOING, A. C. Mortalidade de pedestres em acidentes de trânsito no brasil: análise de tendência temporal, 1996-2015. *Epidemiologia e Serviços de Saúde,* SciELO Public Health, v. 28, p. e2018079, 2019. Citado na página 1.
- GIRSHICK, R. Fast r-cnn. In: *Proceedings of the IEEE international conference on computer vision.* [S.l.: s.n.], 2015. p. 1440–1448. Citado na página 24.
- 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. Citado na página 23.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning.* [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado 4 vezes nas páginas 17, 18, 19 e 20.
- HE, K. et al. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence,* IEEE, v. 37, n. 9, p. 1904–1916, 2015. Citado 2 vezes nas páginas 24 e 29.
- 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. Citado na página 28.
- HUANG, G. et al. Condensenet: An efficient densenet using learned group convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* [S.l.: s.n.], 2018. p. 2752–2761. Citado na página 29.
- IZE, M. d. C. J. Multiple pedestrian tracking using geometric and deep features. 2019. Citado na página 33.
- JAN, Z. et al. A convolutional neural network based deep learning technique for identifying road attributes. In: IEEE. *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ).* [S.l.], 2018. p. 1–6. Citado na página 1.
- JIANG, Z. et al. Real-time object detection method based on improved yolov4-tiny. *arXiv preprint arXiv:2011.04244*, 2020. Citado 2 vezes nas páginas 31 e 32.
- JIM. *Obs Studio.* <https://obsproject.com/pt-br>: [s.n.], 2022. Citado na página 34.

- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, v. 25, 2012. Citado na página 21.
- LECUN, Y. et al. Generalization and network design strategies. *Connectionism in perspective*, North Holland, v. 19, n. 143-155, p. 18, 1989. Citado na página 17.
- LIN, T.-Y. et al. Feature pyramid networks for object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 2117–2125. Citado na página 24.
- LIN, T.-Y. et al. Focal loss for dense object detection. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2980–2988. Citado na página 24.
- LIU, S. et al. Path aggregation network for instance segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2018. p. 8759–8768. Citado na página 29.
- LIU, W. et al. Ssd: Single shot multibox detector. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 21–37. Citado na página 24.
- MARQUES, B. H. P. *Avaliação de algoritmos baseados em deep learning para localizar placas veiculares brasileiras em ambientes complexos*. Dissertação (B.S. thesis) — Brasil, 2019. Citado 2 vezes nas páginas 43 e 44.
- MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 4.
- MCDERMOTT, J. R1: A rule-based configurer of computer systems. *Artificial intelligence*, Elsevier, v. 19, n. 1, p. 39–88, 1982. Citado na página 5.
- MICHELucci, U. *Advanced applied deep learning: convolutional neural networks and object detection*. [S.l.]: Springer, 2019. Citado na página 22.
- MITCHELL, T. M. et al. *Machine learning*. [S.l.]: McGraw-hill New York, 1997. Citado na página 6.
- NIELSEN, M. A. *Neural networks and deep learning*. [S.l.]: Determination press San Francisco, CA, USA, 2015. v. 25. Citado na página 20.
- RASCHKA, S.; MIRJALILI, V. *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-learn, and TensorFlow*. [S.l.]: Packt Publishing, 2017. (Expert insight, isbn:9781787125933). Citado 4 vezes nas páginas 10, 11, 13 e 14.
- 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. Citado 3 vezes nas páginas 24, 26 e 27.
- REDMON, J.; FARHADI, A. Yolo9000: better, faster, stronger. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 7263–7271. Citado 2 vezes nas páginas 27 e 28.

- REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. Citado 2 vezes nas páginas 28 e 29.
- RIBEIRO, H. A. S.; CALHÁO, F. M. Faixas de travessia de pedestre: Proposta de traffic calming para redução de conflitos1. 2017. Citado na página 1.
- RICH, E.; KNIGHT, K. *Artificial Intelligence*. [S.l.]: McGraw-Hill Education, 1991. Citado na página 4.
- ROBOFLOW. *Roboflow*. <https://roboflow.com/>: [s.n.], 2022. Citado na página 35.
- ROSA, J. *Fundamentos da Inteligência Artificial*. [S.l.]: LTC, 2011. Citado 2 vezes nas páginas 4 e 6.
- RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. [S.l.]: Prentice Hall, 2010. Citado na página 5.
- RUSSELL, S.; NORVIG, P. *Inteligência Artificial*. [S.l.]: Rio de Janeiro: Campus Elsevier, 2013. Citado 3 vezes nas páginas 4, 5 e 10.
- SCHALKOFF, R. *Artificial Intelligence: An Engineering Approach*. [S.l.]: McGraw-Hill, 1990. Citado na página 4.
- SONG, H. et al. Vulnerable pedestrian detection and tracking using deep learning. In: IEEE. *2018 International Conference on Electronics, Information, and Communication (ICEIC)*. [S.l.], 2018. p. 1–2. Citado na página 33.
- SOUZA, S. S. et al. Fatores associados aos óbitos por acidentes de trânsito nas rodovias federais da bahia. *Revista Saúde. com*, v. 18, n. 2, 2022. Citado na página 1.
- TOMÈ, D. et al. Deep convolutional neural networks for pedestrian detection. *Signal processing: image communication*, Elsevier, v. 47, p. 482–489, 2016. Citado na página 32.
- TZUTALIN. *LabelImg*. <https://github.com/tzutalin/labelImg>: [s.n.], 2022. Citado na página 36.
- VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: SN. *Proceedings of the xxix conference on graphics, patterns and images*. [S.l.], 2016. v. 1, n. 4. Citado na página 33.
- VASILEV, I. et al. *Python Deep Learning: Exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow*. [S.l.]: Packt Publishing, 2019. ISBN 9781789349702. Citado 5 vezes nas páginas 11, 12, 13, 14 e 20.
- VIDEOLAN. *VLC media player*. <https://www.videolan.org/>: [s.n.], 2022. Citado na página 34.
- WANG, L. et al. An improved light-weight traffic sign recognition algorithm based on yolov4-tiny. *IEEE Access*, IEEE, v. 9, p. 124963–124971, 2021. Citado na página 32.
- ZHANG, L. et al. Cooperative collision warning system design at intersections based on trajectory prediction and conflict risk evaluation. *International Journal of Vehicle Design*, Inderscience Publishers (IEL), v. 87, n. 1-4, p. 95–119, 2021. Citado na página 2.

ZHANG, S. et al. Prediction of pedestrian crossing intentions at intersections based on long short-term memory recurrent neural network. *Transportation research record*, SAGE Publications Sage CA: Los Angeles, CA, v. 2674, n. 4, p. 57–65, 2020. Citado na página 1.

ZOU, Z. et al. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019. Citado na página 23.