

**UNIVERSIDADE FEDERAL DO PAMPA  
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO**

**ANGELO RODRIGUES SILVA**

**REDE NEURAL CONVOLUCIONAL  
APLICADA NO MONITORAMENTO  
POR CÂMERAS DE SEGURANÇA**

**Bagé  
2022**

**ANGELO RODRIGUES SILVA**

**REDE NEURAL CONVOLUCIONAL  
APLICADA NO MONITORAMENTO  
POR CÂMERAS DE SEGURANÇA**

Trabalho de Conclusão de Curso apresentado  
ao curso de Bacharelado em Engenharia de  
Computação como requisito parcial para a  
obtenção do grau de Bacharel em Engenharia de  
Computação.

Orientador: Gerson Alberto Leiria Nunes

**Bagé  
2022**

Ficha catalográfica elaborada automaticamente com os dados fornecidos  
pelo(a) autor(a) através do Módulo de Biblioteca do  
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

S584r SILVA, ANGELO RODRIGUES

REDE NEURAL CONVOLUCIONAL APLICADA NO MONITORAMENTO POR  
CÂMERAS DE SEGURANÇA / ANGELO RODRIGUES SILVA.

68 p.

Trabalho de Conclusão de Curso(Graduação)-- Universidade  
Federal do Pampa, ENGENHARIA DE COMPUTAÇÃO, 2022.

"Orientação: Gerson Alberto Leiria Nunes".

1. Inteligência Artificial. 2. Reconhecimento de Imagens.  
3. Armas. 4. Aprendizagem Profunda. 5. Redes Neurais. I.  
Título.



SERVIÇO PÚBLICO FEDERAL  
MINISTÉRIO DA EDUCAÇÃO  
Universidade Federal do Pampa

**ANGELO RODRIGUES SILVA**

**REDE NEURAL CONVOLUCIONAL  
APLICADA NO MONITORAMENTO  
POR CÂMERAS DE SEGURANÇA**

Trabalho de Conclusão de Curso  
apresentado ao curso de Bacharelado em  
Engenharia de Computação como requisito  
parcial para a obtenção do grau de Bacharel  
em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 10 de Agosto de 2022.

Banca examinadora:

---

Prof. Dr. Gerson Alberto Leiria Nunes  
Orientador – Universidade Federal do Pampa

---

Prof. Dr. Fábio Luis Livi Ramos  
Universidade Federal do Pampa

---

Prof. Dr. Milton Roberto Heinen  
Universidade Federal do Pampa



Assinado eletronicamente por **MILTON ROBERTO HEINEN, PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/08/2022, às 19:25, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.



Assinado eletronicamente por **GERSON ALBERTO LEIRIA NUNES, PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/08/2022, às 19:53, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.



Assinado eletronicamente por **FABIO LUIS LIVI RAMOS, PROFESSOR DO MAGISTERIO SUPERIOR**, em 20/08/2022, às 10:00, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.



A autenticidade deste documento pode ser conferida no site  
[https://sei.unipampa.edu.br/sei/controlador\\_externo.php?acao=documento\\_conferir&id\\_orgao\\_acesso\\_externo=0](https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0), informando o código verificador **0902003** e o código CRC **03E4D9D0**.

Referência: Processo nº 23100.017409/2022-56 SEI nº 0902003

## RESUMO

Considerando a segurança um fator primário para uma boa qualidade de vida e considerando a atual situação em nossa sociedade, torna-se evidente a necessidade de desenvolvermos métodos mais eficientes no combate à criminalidade. Uma das principais medidas para melhorar a segurança é o monitoramento por vídeo, aproveitando-se do avanço em *software*, como nos detectores de objetos, e *hardware*, com o desenvolvimento e difusão de melhores câmeras de vídeo e placas gráficas, necessárias para o alto processamento paralelo das redes neurais. Portanto este projeto tem como proposta o desenvolvimento de um sistema capaz de identificar pessoas portando armas através de imagens de circuito fechado de televisão. Através de uma pesquisa concluiu-se que o estado da arte em detecção de objetos é a série YOLO. Existem diversos estudos sobre a utilização de redes neurais com o mesmo propósito desse projeto, porém diferencia-se dos demais por utilizar sua versão 5 da YOLO. O *database* é composto por imagens de pistola e revólver, sendo criada uma classe única denominada arma. O desempenho da rede é medido através de seus resultados de *precision* 97,58%, *recall* 91,75% e mAP (*mean average precision*) 94,57%. Esta performance foi atingida utilizando um modelo *smal* do YOLOv5 e um *dataset* com 4369 imagens com imagens próprias.

**Palavras-chave:** Inteligência Artificial; Redes Neurais; Aprendizagem Profunda; Redes Neurais Convolucionais. Reconhecimento de Imagens. Circuito Fechado de Televisão. Armas. Detecção.

## ABSTRACT

As security is a primary factor for a good quality of life and considering the current situation in our society, the possibility of developing more efficient tools to deal with insecurity is evident. One of the main measures to improve security is video monitoring, taking advantage of advances in software, such as object detectors, and hardware, with the development and dissemination of better video cameras and graphics cards, necessary for high processing of parallel neural networks. Therefore, this project proposes the development of a system capable of identifying people carrying weapons through closed-circuit television images. Based on a research it was concluded that the state of the art in object detection is the YOLO series. There are several studies on the use of neural networks with the same purpose of this project, but it differs from the others because it uses YOLO version 5. The database is composed of images of pistols and revolvers, being created a unique class called weapon. Network performance is measured through its results of precision 97.58%, recall 91.75% and mAP (mean average precision) 94.57%. This performance was achieved using a YOLOv5 small model and a dataset with 4369 images with own images.

**Keywords:** Artificial Intelligence, Deep Learning, Convolucional Neural Network, Detection, Weapons, YOLOv5 e CCTV.

## LISTA DE FIGURAS

Figura 1	Visão Computacional x Computação Gráfica.....	19
Figura 2	Visão Computacional e disciplinas relacionadas.....	20
Figura 3	Modelo neurônio, de McCulloch e Pitts.....	21
Figura 4	Uma operação de convolução.....	24
Figura 5	Extração de características.....	25
Figura 6	Um exemplo de <i>padding</i> .....	27
Figura 7	Um exemplo de <i>stride</i> .....	28
Figura 8	Um exemplo de um <i>max pooling</i> .....	30
Figura 9	Um exemplo de uma rede <i>fullyconnected</i> .....	31
Figura 10	Ilustração PANet e seus componentes.....	36
Figura 11	Desempenho dos diferentes modelos do YOLOv5.....	36
Figura 12	Ilustração da arquitetura do YOLOv5.....	37
Figura 13	Imagen da classe A na esquerda e da classe B na direita.....	39
Figura 14	Diferentes posições com a arma.....	39
Figura 15	Diferentes intensidades de luz.....	40
Figura 16	Técnicas de aumento do <i>database</i> .....	40
Figura 17	Arquitetura do sistema.....	41
Figura 18	Segmentação e descarte de áreas desnecessárias.....	41
Figura 19	Arquitetura VGG Net.....	42
Figura 20	Arquitetura ZF Net.....	43
Figura 21	Imagens da classe não pistola.....	44
Figura 22	Exemplo de identificação de arma pela RN.....	47
Figura 23	Exemplo de imagem de fuzil modelo AR do <i>database</i> do site <i>Internet Movie Firearms database</i> .....	48
Figura 24	Exemplo de imagem analisada pela RN treinada com o <i>dataset</i> inicial.....	49
Figura 25	Na imagem à esquerda observa-se uma foto com uma câmera de 48 pixels e à direita um recorte realizado pela RN de um vídeo.....	50
Figura 26	À esquerda pessoa portando um controle remoto e à direita portando uma pistola.....	51
Figura 27	Pessoa armada de costas.....	52
Figura 28	Pessoa armada a 90° com contexto limpo.....	52
Figura 29	Pessoa armada mirando para a câmera.....	53
Figura 30	Pessoa armada parcialmente de costas para câmera.....	53
Figura 31	Pessoa armada parcialmente de frente para câmera.....	54
Figura 32	Objeto com contexto de cor semelhante a ele.....	54
Figura 33	Objeto com contexto que gera confusão.....	55
Figura 34	Hiperparâmetros para o treino.....	56
Figura 35	Arquitetura <i>Small</i> .....	57
Figura 36	Arquitetura <i>Medium</i> .....	57
Figura 37	Comparativo entre modelo <i>small</i> na esquerda e <i>medium</i> na direita.....	58
Figura 38	Gráfico sobre o treinamento no <i>dataset</i> ampliado.....	59
Figura 39	Comparativo entre modelo <i>small</i> na esquerda e <i>medium</i> na direita.....	59
Figura 40	Gráfico sobre o treinamento no <i>dataset</i> ampliado.....	60
Figura 41	Comparativo entre modelo <i>small</i> na esquerda e <i>medium</i> na direita.....	60
Figura 42	Gráfico sobre o treinamento no <i>dataset</i> ampliado.....	61
Figura 43	Comparativo entre modelo <i>small</i> na esquerda e <i>medium</i> na direita.....	61
Figura 44	Gráfico sobre o treinamento no <i>dataset</i> ampliado.....	62
Figura 45	Comparativo entre modelo <i>small</i> na esquerda e <i>medium</i> na direita.....	62

Figura 46 Gráfico sobre o treinamento no <i>dataset</i> ampliado.....	63
Figura 47 Comparativo entre modelo <i>small</i> na esquerda e <i>medium</i> na direita. ....	63
Figura 48 Gráfico sobre o treinamento no <i>dataset</i> ampliado.....	64
Figura 49 Comparativo entre modelo <i>small</i> na esquerda e <i>medium</i> na direita. ....	64
Figura 50 Gráfico sobre o treinamento no <i>dataset</i> ampliado.....	65

## **LISTA DE TABELAS**

Tabela 1 Resultado dos classificadores no <i>dataset1</i> .....	44
Tabela 2 Resultado dos classificadores no <i>dataset 2</i> .....	45
Tabela 3 Resultado dos detectores de objetos no <i>dataset 3</i> , IoU <i>threshold</i> de 50%.....	45
Tabela 4 Comparativo de técnicas e <i>framework</i> usados.....	46
Tabela 5 Comparativo de técnicas e <i>framework</i> usados.....	65
Tabela 6 Comparativo de trabalhos e seus resultados. ....	66

## **LISTA DE ABREVIATURAS E SIGLAS**

ACM	Association for Computing Machinery
BiFPN	Weighted Bi-directional Feature Pyramid Network
CCTV	Closed Circuit Television
CSPNet	Cross Stage Partial Networks
FPN	Feature Pyramid Network
FPS	Frames Per Second
GPU	Graphical Processing Unit
IA	Inteligência Artificial
IoT	Internet of Things
mAP	mean Average Precision
OCR	Optical Character Recognition
PANet	Path Aggregation Network
ReLU	Rectified Linear Unit
RN	Rede Neural
RNA	Rede Neural Artificial
RNC	Rede Neural Convolucional
RGB	Red Green Blue
SGD	Stochastic Gradient Descent
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
VGG	Visual Geometry Group
GFLOPS	Giga Float Point Operations Per Second

## SUMÁRIO

<b>1 INTRODUÇÃO .....</b>	<b>12</b>
<b>1.1 Problema.....</b>	<b>13</b>
<b>1.2 Objetivos .....</b>	<b>13</b>
<b>1.2.1 Objetivo Geral.....</b>	<b>13</b>
<b>1.2.2 Objetivos Específicos .....</b>	<b>13</b>
<b>1.3 Metodologia .....</b>	<b>14</b>
<b>1.4 Organização do trabalho .....</b>	<b>15</b>
<b>2 REVISÃO BIBLIOGRÁFICA.....</b>	<b>17</b>
<b>2.1 Inteligência Artificial .....</b>	<b>17</b>
<b>2.2 Visão Computacional.....</b>	<b>18</b>
<b>2.3 Redes Neurais .....</b>	<b>20</b>
<b>2.3.1 Deep Learning .....</b>	<b>22</b>
<b>2.4 Redes Neurais Convolucionais .....</b>	<b>23</b>
<b>2.4.1 Padding .....</b>	<b>27</b>
<b>2.4.2 Pooling .....</b>	<b>29</b>
<b>2.4.3 Treinamento e aprendizado.....</b>	<b>31</b>
<b>2.4.4 YOLOv5.....</b>	<b>34</b>
<b>3 TRABALHOS CORRELATOS .....</b>	<b>38</b>
<b>3.1 Modelos convolucionais na detecção de armas. ....</b>	<b>38</b>
<b>3.2 Comparativo de diversas arquiteturas e técnicas.....</b>	<b>42</b>
<b>3.3 IoT Based Weapons Detection System .....</b>	<b>45</b>
<b>4 CRIAÇÃO DO DATASET E TREINAMENTO .....</b>	<b>47</b>
<b>4.1 Dataset Inicial .....</b>	<b>48</b>
<b>4.2 Dataset Próprio .....</b>	<b>50</b>
<b>4.3 Resultados.....</b>	<b>58</b>
<b>5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS.....</b>	<b>66</b>
<b>REFERÊNCIAS .....</b>	<b>67</b>

## 1 INTRODUÇÃO

Existem diversos sistemas de segurança, como alarmes com sensor de presença, alarmes em portas e janelas, rondas armadas ou não, entre outros. Porém, atualmente um dos métodos de se aumentar a segurança em um local é a implantação de câmeras de segurança, que possibilitam o monitoramento em tempo real e também a análise posterior das imagens para resolução de crimes ou qualquer evento não explicado. Outro fator relevante é a necessidade de um profissional treinado e atento para executar a tarefa de monitoramento, exigindo uma equipe que dividirá os turnos possibilitando um monitoramento 24 horas por dia, podendo tal profissional sentir o peso de diversas horas de trabalho, sofrer de fadiga e acabar perdendo desempenho em sua tarefa de monitoramento.

A sensação de insegurança nos obriga a mantermos um estado de alerta constante, porém, esse estilo de vida é extremamente desgastante. Considerando uma pessoa com o intuito natural de se defender, deve contar no mínimo com uma estrutura protetora que forneça cobertura e abrigo dificultando um engajamento inicial não esperado por algum atacante. Além disso, deve haver alguma ferramenta ou armamento capaz de cessar tal ataque. Entretanto, um dos problemas é a dificuldade em estabelecer e manter um sistema composto por homens dispostos em pontos estratégicos com visão para qualquer possível hostilidade. Portanto, nesse caso hipotético apenas existe um homem responsável por sua segurança, então uma das medidas mais coerentes seria a instalação de um sistema de vigilância baseado em um circuito fechado de televisão. Então, considera-se a existência de três pontos a serem observados que fornecem segurança ao indivíduo, seriam três telas que exigiram atenção constante. Trabalho esse que necessita de uma equipe treinada, algo que está fora do escopo de nossa hipótese.

Com o avanço tecnológico em *hardware* e *software*, as redes neurais convolucionais passaram a demonstrar ser uma ótima ferramenta na tarefa de classificar imagens se mostrando promissora para a automação da detecção de pessoas armadas, usando câmeras de segurança, em locais considerados sensíveis. Portanto, se faz necessário o desenvolvimento de um sistema capaz de identificar pessoas em locais e/ou horários não permitidos, bem como a detecção de armas de fogo, tais como: pistola, revólver, espingarda e fuzil. A fim de realizar esta tarefa existem diversas tecnologias, mas a que se destaca é a rede neural convolucional, mais especificadamente nesse projeto utiliza-se o YOLOv5 na identificação de pessoas em imagens de câmeras de segurança,

tendo em vista sua alta velocidade de operação possibilitando seu emprego em tempo real. Após identificar que há uma ou mais pessoas em uma imagem, realiza-se o recorte das regiões de interesse identificadas nesta imagem. Estes recortes são repassados para uma segunda rede neural treinada para identificar determinados tipos de armas.

## 1.1 Problema

Visando a automação dos alertas para forças de segurança quando identificadas armas em imagens de vigilância por câmeras e levando em consideração as limitações de *hardware* e *software*, como também a dificuldade de identificação de objetos pequenos. É possível desenvolver um sistema computacional baseado em redes neurais convolucionais capaz de identificar pessoas armadas a partir de imagens de câmeras de segurança?

## 1.2 Objetivos

Nesta seção são descritos os objetivos dessa pesquisa. Primeiramente descreve-se o objetivo geral que deve ser alcançado ao final da realização do trabalho. Além disso, os objetivos específicos são descritos em forma de etapas, as quais devem realizadas para o cumprimento do objetivo geral.

### 1.2.1 Objetivo Geral

Desenvolver sistema computacional baseado em uma rede neural convolucional para identificar indivíduos armados através de imagens de câmeras de segurança para automatizar o processo de monitoramento e detecção de invasores hostis e otimizar tomada de decisão na conduta da força de reação assim diminuindo os custos de pessoal e também o risco a vida iminentes a tais atividades de ronda e segurança.

### 1.2.2 Objetivos Específicos

- Realizar revisão bibliográfica sobre inteligência artificial, redes neurais, aprendizagem profunda, redes neurais convolucionais, YOLOv5, treinamento e

validação.

- Entender e testar o funcionamento do YOLOv5, suas opções de arquiteturas e configurações de treinamento para imagens, vídeos e webcam.
- Criar *dataset* (conjunto de dados) com imagens de pessoas portando armamento a partir de câmeras de segurança para serem usadas no treinamento, validação e testes.
- Classificar as imagens do *dataset* destacando os objetos a serem identificados.
- Treinar rede neural convolucional Darknet, YOLOv5, utilizando 4/5 do *dataset* para identificar pessoas armadas.
- Executar ajustes na RNC (Rede Neural Convolutional) para otimizar os resultados.
- Validar o modelo treinado, através da análise do mAP e dos fps(frames por segundo).
- Analisar dados gerado e construir gráficos a cerca dos resultados obtidos.
- Exportar resultados e considerações finais.

### 1.3 Metodologia

Inicialmente realiza-se uma pesquisa bibliográfica a cerca do assunto culminando na escolha da arquitetura utilizada. Posteriormente desenvolvimento do projeto experimental buscando um bom desempenho.

O método adotado é o de revisão sistemática da literatura, na busca do conhecimento necessário para uma pesquisa científica adequada. Avaliar os trabalhos atuais sobre o tema e sua relevância.

Para isso algumas questões são levantadas. Foram auferidas algumas questões importantes para definir a pesquisa, são elas por sua ordem:

- Quais técnicas de IA que vêm sendo bem sucedidas na detecção de pessoas armadas?
- Qual a arquitetura estado da arte de Rede Neural Convolutional usada para esses tipos de detecção?
- O tipo de arma ou cenário, pode interferir na precisão da identificação?

Para a definição das palavras-chave foram observadas as questões de pesquisa e o título do trabalho. Dando continuidade a nossa revisão sistemática, prosseguimos com a montagem de uma String de busca. Com a combinação das palavras-chave levantadas e operadores lógicos, chegando no seguinte resultado: (identificação OR detecção OR reconhecimento) AND (armas OR armas de fogo OR armamento) AND (imagem OR câmeras de segurança OR circuito-fechado de televisão). ou em inglês: (*identification OR detection OR recognition*) AND (*gun OR gunfire OR weapons*) AND ("*surveillance camera*"OR *image* OR "*security cameras*"OR "*closed circuit television*")

Tanto em inglês como em português, após montada as Strings de busca, elas foram colocadas nas principais bases eletrônicas que indexam pesquisas produzidas na área de Ciência da Computação, são elas:

- Scopus <sup>1</sup>
- IEEE Xplore <sup>2</sup>
- ScienceDirect <sup>3</sup>
- Springer <sup>4</sup>
- Web of Science <sup>5</sup>
- ACM <sup>6</sup>
- Compendex <sup>7</sup>
- Google Scholar <sup>8</sup>

Os artigos encontrados foram avaliados de acordo com a contemporaneidade e técnicas relevantes ao projeto experimental.

#### **1.4 Organização do trabalho**

Este trabalho está organizado em cinco capítulos. No capítulo 1 foi apresentada a introdução do projeto que também é compreendida pela definição do problema, a apresentação do objetivo geral e dos objetivos específicos e também apresentação da

---

<sup>1</sup>(www.scopus.com)

<sup>2</sup>(www.ieeexplore.com)

<sup>3</sup>(www.sciencedirect.com)

<sup>4</sup>(www.springerlink.com)

<sup>5</sup>(www.isiknoledge.com)

<sup>6</sup>(www.portal.acm.org)

<sup>7</sup>(www.engineeringvillage.com)

<sup>8</sup>(scholar.google.com)

motivação para a execução do projeto. No capítulo 2 são apresentados os seguintes tópicos: caracterização da pesquisa, métodos e procedimentos adotados. No capítulo 3 são abordados os seguintes tópicos: conceitos básicos sobre inteligência artificial, visão computacional, *deep learning*, redes neurais, redes neurais convolucionais, o *software YOLOv5*, noções sobre *dataset*, métodos de aprimoramento do treinamento e do aprendizado e por fim, como será feita a validação do modelo proposto. No capítulo 4 abordaremos diversos trabalhos que se assemelham por terem os objetivos ou os métodos similares ao do presente projeto, destacando seus prós e contras, também fazendo um comparativo. Por fim, no capítulo 5 são apresentadas as considerações finais e trabalhos futuros.

## 2 REVISÃO BIBLIOGRÁFICA

### 2.1 Inteligência Artificial

A definição de inteligência não é algo trivial na comunidade científica, segundo o psicólogo e psico-metrista Sternberg (2010), define-se inteligência como "a capacidade do ser humano tem para aprender com a experiência, usando processos metacognitivos para incrementar a aprendizagem e a capacidade de adaptação ao ambiente que nos cerca".

De acordo com o matemático e cientista da computação Minsky. (1986), que é um dos fundadores do campo da IA, o correto seria ter uma posição oposta a querer definir inteligência e deveríamos sim buscar explicar como a usamos. Segundo ele "nossas mentes contém processos efetuados por meio de agentes que nos capacitam a resolver problemas que consideramos difíceis. Inteligência é o nome que damos a qualquer um destes processos que ainda não compreendemos".

Dentre outros pesquisadores, o psicólogo Howard Gardner entende que a inteligência corresponde à capacidade de resolver problemas ou de elaborar produtos que sejam importantes para um determinado ambiente ou comunidade cultural específica. Conforme Gardner e Veronese (1995) "A capacidade de resolver permite à pessoa abordar uma situação em que um objetivo deve ser atingido e localizar a rota adequada para alcançar este objetivo."

Segundo Medeiros (2018), podemos observar o desenvolvimento de duas linhas fundamentais de pesquisa oriundas da convergência de diversas disciplinas com abordagens variadas. Se assumindo a perspectiva que prioriza o cérebro com seus elementos mais básicos, os neurônios e as sinapses que os conectam, nosso principal desejo é construir cérebros artificiais, com neurônios artificiais que simulem a maneira como os mecanismos eletroquímicos dos cérebros biológicos funcionam, mas quando adota-se uma perspectiva centrada na mente, que atua com base no pensamento de símbolos, como um *software* executado sobre um *hardware* (cérebro fisiológico), então nosso desejo mais decisivo é construir dispositivos que lidem com tais símbolos da mesma forma como a mente humana os processa. Atualmente temos três grandes linhas de pesquisa sobre IA, uma linha simbólica, uma conexionista e uma evolucionária.

A linha conexionista é voltada para a construção de arquitetura de dispositivos que simulem as células biológicas que interagem para que ocorram os processos responsáveis pela inteligência. Onde destacam-se as redes neurais artificiais. Segundo Haykin (2007),

as redes neurais artificiais constituem um campo de pesquisa que tem por preocupação lidar com tarefas como o reconhecimento de padrões, a previsão e a tomada de decisão mediante o uso de redes de unidades conectadas, treinadas por algoritmos que funcionam com base em amostras do mundo real e podem, assim, aprender a classificar padrões.

Na linha de pesquisa simbólica o foco é lidar com processos inteligentes utilizando linguagens baseadas em lógica e na construção de redes semânticas para solucionar problemas e simular conhecimento especialista para contextos de diagnóstico. Esta linha é constituída de sistemas baseados em conhecimento, incluindo as linhas de pesquisa sobre a linguagem Lisp (BITTENCOURT, 1998), que trabalha com representação do conhecimento em forma de listas, e a linguagem de programação Prolog (PALAZZO, 1997), que permite a manipulação de símbolos através de representação de conhecimentos na forma de fatos e regras.

De acordo com Norvig (2013), os sistemas especialistas constituem uma das áreas mais relevantes da linha simbólica. O termo se refere a sistemas em que o conhecimento de um especialista humano em uma área bem delimitada é representado por uma linguagem, de forma a permitir o diagnóstico de situações e a execução de ações como se um ser humano os fizesse. Portanto, é importante salientar que na linha simbólica, a preocupação é dirigida à forma como a mente pensa e não no funcionamento do cérebro, com suas partes e divisões. Ainda na área simbólica, existem pesquisas sobre ontologias, que se referem a representações do conhecimento obtido por consenso.

A linha evolucionária, também chamada de computação evolucionária é um ramo de pesquisa que busca soluções inspiradas na natureza, analisando a competição das espécies. Tem como principais estudos os algoritmos genéticos e a programação genética, que buscam estudar fenômenos relacionados às adaptações das espécies e como incorporar tais conceitos na computação.

## **2.2 Visão Computacional**

A visão computacional é um campo da ciência da computação que visa dar aos computadores a capacidade de processar e identificar imagens da mesma forma dos seres humanos. Segundo Bhuyan (2019), tenta-se imitar o sistema visual humano através de sistemas artificiais capazes de extrair informações de imagens possibilitando um maior entendimento sobre a situação. Os dados de entrada podem ser vídeos, imagens profundas ou dados multidimensionais de sensores de imagem. A visão computacional

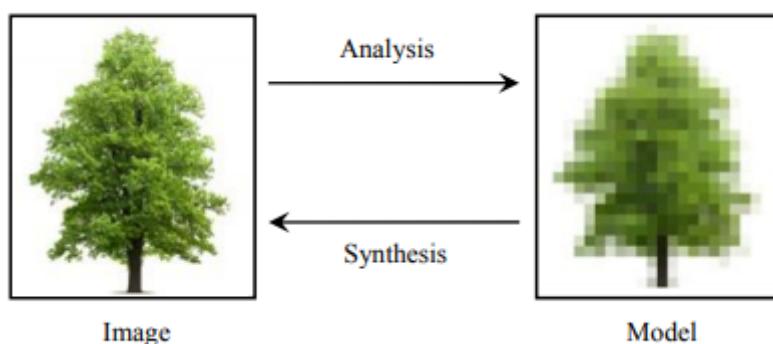
tem como objetivo principal descrever uma cena do mundo real em uma ou mais imagens e identificar e reconstruir suas propriedades, como características de cor, informações sobre formato, características de textura, iluminação, entre outras.

Conforme a figura 1 observa-se a semelhança entre um sistema visual humano e o sistema visual do computador. Ambos os sistemas têm com o mesmo princípio de funcionamento, ou seja, a conversão de luz em informação útil a partir da qual modelos precisos do mundo físico são construídos em ambientes computacionais capazes dessa simulação. Da mesma forma, em um nível mais elevado os dois sistemas são similares. Pois ambos possuem sensores de luz que converte fótons em sinais, uma etapa de processamento e por fim um mecanismo de interpretação definida por uma etapa de reconhecimento de objeto. A diferença entre visão computacional, processamento de imagem e computação gráfica pode ser resumida assim:

- Na visão computacional (análise e interpretação da imagem e compreensão da cena) a entrada é uma imagem e a saída é a interpretação da cena. A análise de imagem se preocupa em fazer medições quantitativas da imagem para dar uma descrição dela.
- No processamento de imagens (recuperação, filtragem, compressão, visualização de imagens) a entrada e a saída são compostas por imagens.
- Na computação gráfica a entrada é uma cena do mundo real e a saída é uma imagem.

A visão computacional cria modelos a partir da análise de imagens. Por outro lado, a computação gráfica através da síntese de um modelo como entrada e convertendo em uma imagem. Considerando que a visão computacional lida com a construção de máquinas que podem ver e interagir com o mundo como demonstrado na figura 1.

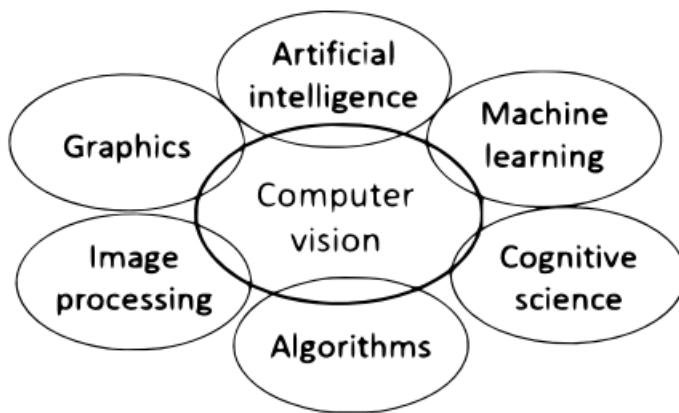
Figura 1 – Visão Computacional x Computação Gráfica



Fonte:(BHUYAN, 2019).

As aplicações no mundo real para a visão computacional são inúmeras, pode-se destacar as seguintes: inspeção de máquina, reconhecimento ótico de caracteres, Optical Character Recognition (OCR), construção de modelos em três dimensões, análise de imagens médicas, automação na vigilância por video, biometria, fusão e costura de imagens, metamorfose, entre outras. A figura 2 mostra as diversas pesquisas que estão relacionadas à visão computacional.

Figura 2 – Visão Computacional e disciplinas relacionadas



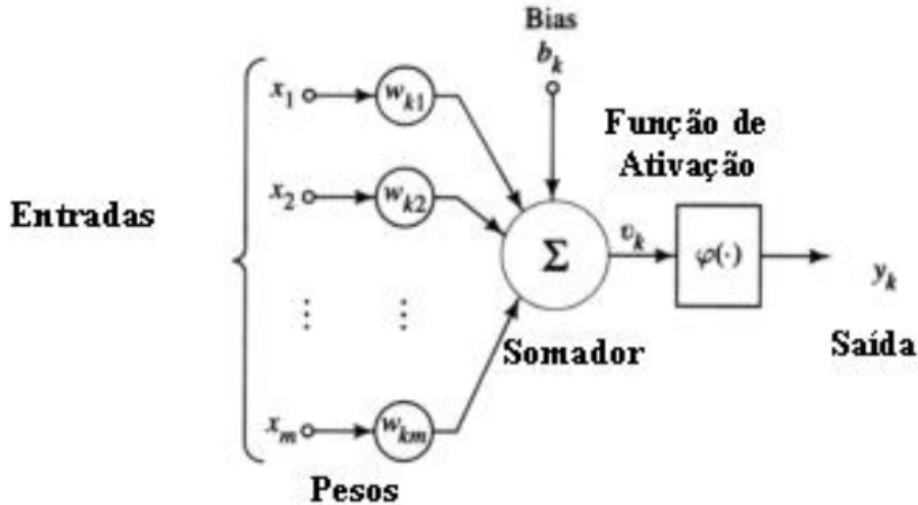
Fonte:(BHUYAN, 2019).

### 2.3 Redes Neurais

As redes neurais artificiais, RNA, são modelos matemáticos similares as estruturas neurais biológicas, tendo sua capacidade computacional desenvolvida através de aprendizado e generalizações. O aprendizado é a capacidade das RNA de ajustar seus parâmetros de acordo com a interação com o meio externo, por consequência, melhorando seu desempenho gradativamente. No treinamento, para um determinado conjunto de dados, seus critérios podem estar associados com o erro quadrático médio das respostas. A capacidade de dar respostas coerentes para dados não vistos chama-se generalização. A generalização e o aprendizado estão ligados, sendo a generalização uma consequência do aprendizado. O processamento da informação é executado por elementos processadores simples de maneira paralela e distribuída. Na figura 3 observa-se que o modelo MCP, McCulloch e Pitts, temos um vetor de entrada  $x$ , um vetor de pesos  $W$ , e a saída  $y$  do neurônio também chamada de saída de ativação, que é obtida através da aplicação de uma

função de ativação,  $f(u)$ , podendo assumir diversas formas normalmente não-lineares.

Figura 3 – Modelo neurônio, de McCulloch e Pitts.



Fonte: Adaptado de McCulloch & Pitts (1943).

A RNA é conhecida como aproximadores universais de funções multivariáveis contínuas. Portanto, problemas de aproximação em funções contínuas podem ser resolvidos por meio de RNA, não dependendo do número de variáveis. A dimensionalidade dos dados determina o número de entradas e saídas de uma RNA. Enquanto que a complexidade do problema determine o número de neurônios nas camadas intermediárias. Então, quanto mais neurônios nas camadas intermediárias, mais complexas são as funções mapeadas em uma RNA, todavia neurônios em excesso podem gerar resultados indesejáveis.

A RNA é capaz de resolver diversos tipos de problemas, como aproximações, previsões, classificações, categorizações e otimizações. Algumas das aplicações são reconhecimento de voz, caracteres, previsões de séries temporais, modelagem de processos, controle, entre outras. Os problemas de aproximações e regressões, são categorizados pela interpolação, isto é, dados são fornecidos dentro de determinados limites, onde a função é definida, visando se aproximar dentro dos mesmos o modelo é ajustado. Previsão financeira, previsão de tempo e modelagem de processos dinâmicos industriais são exemplos destes problemas.

Nos problemas de classificação, o objetivo é atribuir um padrão de entrada e uma classe entre um conjunto de classes conhecidas. Reconhecimento de voz, caracteres e imagens. Nestes casos as classes são definidas pelos sons a serem reconhecidos, conjunto de caracteres desejados e imagens ou objetos a serem classificados. A RNA

é caracterizada pelo aprendizado por exemplos. O algoritmo de aprendizado deve ajustar os parâmetros na convergência para a solução, para um determinado conjunto de dados. O processo de aprendizagem é encontrar o ajuste do vetor de pesos  $W$  para atingir a convergência. Além disso, o critério de convergência varia em função do algoritmo e o paradigma de aprendizagem, podendo envolver a minimização de uma função objetivo, a variação do erro de saída ou mesmo a variação das magnitudes dos pesos da rede.

O uso do *bias* é comum em redes neurais, também é possível utilizá-lo nas operações à frente. Cada filtro em uma camada está associado ao seu próprio *bias*. Portanto em cada filtro de cada camada tem um *bias* de formato  $b(p,q)$ . Cada operação na camada  $q$  com o filtro  $p$  será adicionado ao produto escalar o valor de  $b(p,q)$ . O uso do *bias* significa um incremento de apenas 1 parâmetro em cada filtro, portanto não representa uma sobrecarga, *overhead*.

Como todos os outros parâmetros, o *bias* é aprendido durante a retro-propagação. Considerando o *bias* como o peso de conexão cuja entrada é sempre definida como 1. Portanto, o número de recursos de entrada da camada  $q$  é dado pela equação 1.

$$1 + L_q + B_q + d_q \quad (1)$$

### 2.3.1 Deep Learning

Segundo LeCun, Bengio e Hinton (2015), o *deep learning* é uma parte da IA que se dedica ao aprendizado de máquina, tentando modelar abstrações de alto nível de dados, usando várias camadas de neurônios que consistem em estruturas complexas e não lineares. O aumento da quantidade de dados e do poder de computação as redes neurais com estruturas mais complexas têm atraído atenção, sendo aplicada a vários campos. A pesquisa sobre aprendizado profundo tenta modelar dados em grande escala por múltiplas camadas, assim ao contrário das arquiteturas rasas, as arquiteturas de aprendizado profundo são compostas por diversas transformações não lineares.

Existem diversas arquiteturas profundas diferentes, cada uma pode ser usada para um diferente tipo de dado. Por exemplo, uma rede neural convolucional é a arquitetura mais popular para reconhecimento de imagem, e a rede neural recorrente é mais aplicável a tarefas sequenciais, como a escrita à mão ou o reconhecimento de fala. Logo a seguir, será abordada umas das técnicas que estão tirando maior proveito do aprendizado profundo, as redes neurais convolucionais.

## 2.4 Redes Neurais Convolucionais

Uma Rede neural convolucional é um tipo de rede neural *feed-forward* que usa convolução em pelo menos uma de suas camadas. Uma RNC combina redes artificiais e evolução discreta para o processamento de imagens que podem extrair recursos automaticamente. Portanto, sendo especialmente projetada para reconhecer dados bidimensionais, como imagens. As imagens podem ser usadas diretamente como entrada da rede, o que evita o processo complexo de extração de recursos e reconstrução de dados em algoritmos de reconhecimento de imagem tradicionais. (HAO; ZHANG; MA, 2016). Em 1980 Fukushima (1980) propôs um modelo de rede neural para um mecanismo de reconhecimento de padrões visuais, "neocognitron", que é o predecessor da CNN (*Convolucional Neural Network*). Depois disso os pesquisadores tentaram maneiras diferentes para treinar redes neurais multicamadas e as aplicou a vários cenários. No entanto, enquanto a amplitude e a profundidade de uma rede aumentam, o desempenho dos algoritmos de RNC foram limitados por recursos de computação. Depois de 2006, *GPUs* (*Graphical Processing Unit*) eficientes se tornaram dispositivos de computação generalizados, o que tornou possível o treinamento de redes maiores. Assim, vários modelos têm sido propostos para treinar redes neurais convolucionais profundas de forma mais eficiente. Hoje, a CNN é usada em muitas áreas, incluindo compreensão de imagem e vídeo.

Em uma RNC os estados de cada camada são organizados em uma estrutura de grade. Essas relações espaciais são herdadas de uma camada para outra porque cada valor de recurso é baseado em uma pequena região espacial na camada anterior. Estas relações espaciais são importantes para as operações de convolução e transformações para a próxima camada. Vale salientar a diferença entre profundidade de uma camada, que pode ser composta pelas camadas das cores ou do número de mapas de recurso, e profundidade da própria rede, que seria o número de camadas de neurônios. A rede neural convolucional funciona muito como uma rede neural *feed-forward* tradicional, exceto que as operações em suas camadas são organizadas espacialmente com esparsas conexões entre as camadas.

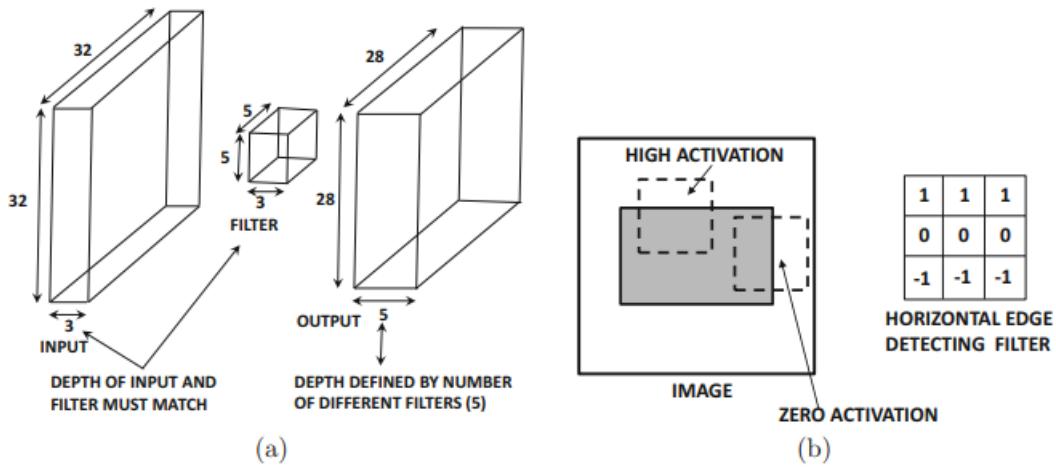
A função de ativação, que pode ser ReLU (Rectified Linear Unit) não é diferente da função de uma rede neural tradicional. Existe também um conjunto final de camadas totalmente conectadas que terminam em um conjunto de nós de saída.

Tendo em vista os dados de entrada de uma CNN ser uma imagem, ela é composta

por uma matriz tridimensional de *pixels* sendo composta por altura, largura e três canais de cores. Pode-se exemplificar com um tamanho de imagem popularmente usado composto por  $32 \times 32$  pixels, tendo três canais de cores RGB, (*Red, Green and Blue*), restando uma configuração  $32 \times 32 \times 3$ .

A operação de convolução aplica o filtro em cada posição possível na imagem ou camada oculta para que o filtro se sobreponha totalmente à imagem e execute um produto escalar entre os parâmetros do filtro e a grade correspondente na entrada. O produto escalar é realizado tratando a entrada tridimensional e os filtros como vetores, sendo ordenados de acordo com seus correspondentes no volume de grade estruturada. O número de alinhamentos entre o filtro e a entrada determinam a altura e largura da próxima camada oculta. Para sabermos o tamanho da próxima camada subtraímos o tamanho do filtro do tamanho da entrada e somaremos um:  $L_q - F_q + 1$ . Tendo uma entrada de tamanho  $32 \times 32$  e um filtro de  $5 \times 5$ , temos na próxima camada um tamanho de  $28 \times 28$  e profundidade da saída é igual ao número de filtros utilizados, como mostra a figura 4.

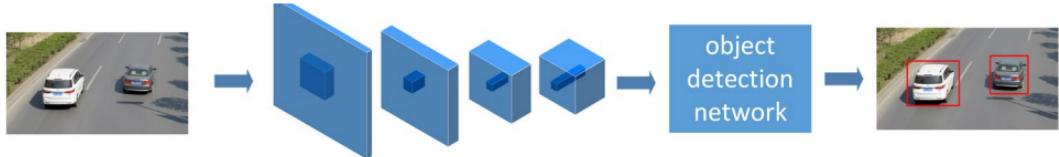
Figura 4 – Uma operação de convolução.



Fonte: (LIU et al., 2018).

Por exemplo, o filtro mostrado na Figura 4 (b) representa um detector de borda horizontal em uma imagem em tons de cinza com um canal. Conforme mostrado na figura 4 (b), o recurso resultante terá alta ativação em cada posição onde uma borda horizontal é vista. Uma borda vertical fornecerá ativação zero, enquanto uma borda inclinada pode fornecer ativação intermediária. Portanto, deslizar o filtro em qualquer lugar da imagem já detectará vários contornos-chave da imagem em um único mapa de recursos. Vários filtros são usados para criar um volume de saída com mais de um mapa de recursos que pode-se ver na figura 5.

Figura 5 – Extração de características.



Fonte: (LI et al., 2019).

Representa-se os dados de entrada tridimensionalmente, porém somente duas dimensões tem relações espaciais e a terceira dimensão é dedicada a propriedades independentes ao lodo dos canais. As intensidades das cores primárias são independentes na primeira camada. Nas camadas ocultas, essas propriedades independentes correspondem a vários tipos de formas extraídas de regiões locais da imagem. Usaremos um formato  $L_q \times B_q \times d_q$ , sendo  $L$  correspondente a altura, o  $q$  a largura e o  $d$  o número de camadas. o valor de  $d_q$  é muito maior que três para as camadas ocultas, porque o número de propriedades independentes de uma determinada região que são relevantes para a classificação pode ser muito significativa.

Para  $q > 1$  chamamos essas grades de valores de mapas de recursos ou mapas de ativação. Em uma CNN os parâmetros são organizados conjuntos de unidades estruturais tridimensionais conhecidas como filtro ou *kernel*. As dimensões do filtro são sempre quadradas e muito menores que as dimensões da camada que ele é aplicado. Porém, a profundidade do filtro sempre será a mesma da camada aplicada. Valores comuns para tamanho de filtro seriam números inteiros pequenos e ímpares.

Nessa outra camada temos uma profundidade de cinco camadas pois estamos usando cinco diferentes filtros cada um com seu conjunto de parâmetros. Cada um desses cinco conjuntos de recursos espacialmente arranjados e obtidos a partir da saída de um único filtro é referido como um mapa de recursos. O número de filtros usados em cada camada controla a capacidade do modelo porque controla diretamente o número de parâmetros. além disso aumentar o número de filtros acaba aumentando o número de mapas de recursos da próxima camada. Na primeira camada temos apenas três canais, porém nas camadas escondidas podemos ter mais de quinhentos.

O filtro é aplicado com o intuito de identificar um tipo particular de padrão espacial em uma pequena região da imagem, portanto, um grande número de filtros é necessário pra identificar uma grande variedade de formas possíveis que combinadas formam as imagens. A camadas posteriores à entrada tendem a perder essa relação espacial, mas

com um número maior de mapas de recursos. Tomamos a figura 4 (b) como exemplo, mostrando um filtro detector de borda horizontal, em uma borda vertical este filtro terá zero ativação, em uma borda inclinada terá média ativação e em uma borda horizontal terá alta ativação.

Portanto, deslizar um filtro em uma imagem detectará vários contornos chave na imagem em um único mapa de recurso. Utiliza-se vários filtros para se obter um volume de saída com mais de um mapa de recurso. Em uma CNN aplica-se o filtro em todas as localizações da imagem para identificar formas. Os filtros das camadas anteriores tendem a buscar formas mais primitivas enquanto que os filtros de camadas posteriores tende a buscar formas complexas formadas pelas formas primitivas.

Uma propriedade da convolução é a equivariância à translação, portanto se for mudado os valores de um pixel da entrada em qualquer direção em uma unidade, em seguida aplicar a convolução, os valores dos recursos correspondente mudarão com os valores de entrada. Pois os parâmetros dos filtros são compartilhados através de toda convolução, e isso se deve ao fato que a presença de uma forma deve ser processada da mesma maneira, independente da sua localização na imagem. Em uma operação de convolução, as contribuições dos produtos escalares sobre todos os mapas de recursos da região da grade correspondente de uma camada precisam ser adicionados para formar um único valor de saída.

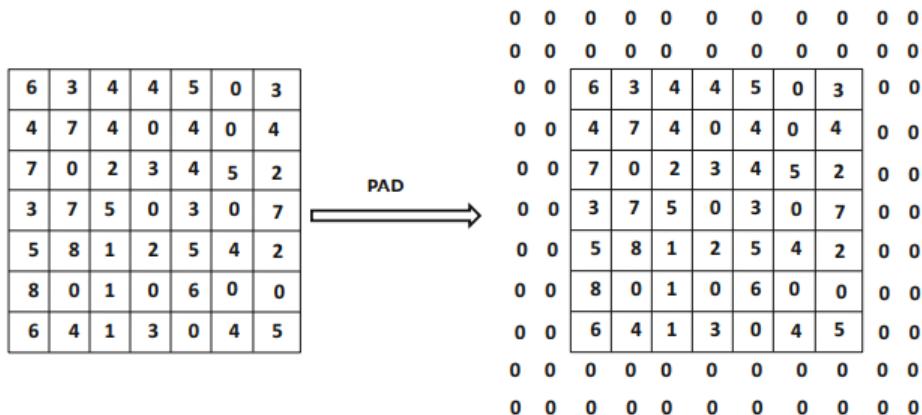
No caso da profundidade da camada e do filtro serem ser dois ou mais, as operações são realizadas para cada mapa espacial e posteriormente agregados em toda a profundidade do filtro. Uma convolução na camada  $q$  aumenta o campo receptivo de um recurso na camada  $q + 1$ . Considerando em uma convolução usando um filtro de  $3 \times 3$  em três camadas consecutivas as ativações capturam *pixels* de região de tamanho  $3 \times 3$ ,  $5 \times 5$  e  $7 \times 7$ , respectivamente, da imagem original de entrada. Outras operações também alcançam o aumento do campo receptivo, diminuindo a pegada espacial das camadas. Isto se deve ao fato que recursos em camadas posteriores capturam características complexas da imagem em regiões espaciais maiores, combinando os recursos mais simples das camadas anteriores. A profundidade de camada  $q + 1$  é igual ao tamanho do filtros da camada  $q$ .

### 2.4.1 Padding

Após uma operação de convolução temos a redução do tamanho da entrada, mas esta ação pode acabar em uma perda de parte da entrada ou do mapa de recursos, se tratando de camadas escondidas, portanto adiciona-se uma borda com cerca de  $(F_q - 1)/2$  pixels com o intuito de resolver tal problema, tal técnica chama-se *padding*. Como consequência a altura e a largura do volume irão aumentar em  $(F_q - 1)$ , que é exatamente o tamanho da redução que a convolução imprime nesta camada. Como estes pixels são preenchidos com valor zero, eles não contribuem para o produto escalar.

Portanto, a operação de *padding* permite a operação de convolução em uma porção do filtro saindo pelas bordas, neste caso chamamos de *half-padding* pois quase metade do filtro fica saindo pelas bordas, no caso do filtro estar posicionado mais extremamente em altura ou largura da camada. Esta operação é utilizada para manter a *footprint*. No caso de não utilização do preenchimento, *padding*, chamamos de *valid padding*, porém esta operação não obtém êxito pois os pixels das bordas são sub-representados em relação aos do centro. tendo em vista a garantia da integridade dos dados, independente da camada, é normalmente usado o *padding*. Observando a figura 5, considerando uma entrada de  $32 \times 32 \times 3$  e um filtro de  $5 \times 5 \times 3$ . Portanto,  $(5 - 1)/2 = 2$  zeros são preenchidos em todos os lados, aumentado para  $36 \times 36 \times 3$  e que posterior a operação de convolução retorna ao tamanho de  $32 \times 32 \times 3$ .

Figura 6 – Um exemplo de *padding*.



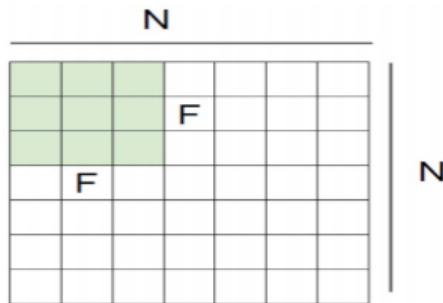
Fonte: (LIU et al., 2018).

O *full-padding* consiste em adicionar  $(f_q - 1)$  zeros em cada lado, portanto, cada

dimensão espacial cresce  $2(F_q - 1)$ . ao executar a convolução a entrada diminui ( $F_q - 1$ ), então após um *full-padding* temos um aumento na *footprint*. Pode-se implementar uma operação de convolução reversa aplicando uma convolução na saída de uma operação de preenchimento total, *full-padding*. Vale ressaltar que o valor que o preenchimento total, *full-padding*, aumenta na pegada espacial é o mesmo que diminui ela no meio-preenchimento, *half-padding*. A convolução reversa é utilizada em algoritmos de retro-propagação e auto-encoder para CNN.

Existem outras técnicas que também visam reduzir a pegada espacial da imagem ou camada oculta. Na abordagem acima, executa-se a convolução sobre todas as posições no espaço localizado no mapa de recursos. Contudo existe o conceito de *strides* ou passadas, que consiste em aumentar o número de deslocamentos entre as convoluções, no abordagem anterior temos uma passada de valor 1, entretanto pode-se aumentar essa passada acarretando em um rápido aumento do campo receptivo de cada recurso. Passadas maiores podem ser úteis em sistemas com memória restrita, ou ainda reduzir o sobre ajuste se a resolução espacial é desnecessariamente alta. Com o pode-se ver na imagem 7.

Figura 7 – Um exemplo de *stride*.



Fonte: (ALBAWI; MOHAMMED; AL-ZAWI, 2017).

Uma configuração típica tem *strides* de valor 1, além disso as imagem costumas ter dimensões quadradas  $L_q = B_q$ . Quando as imagens não são quadradas o pré-processamento é usado para impor isso. O número de filtros em uma camada é frequentemente uma potência de dois, pois geralmente resulta em um processamento mais eficiente. Esta abordagem também leva a profundidade na camada oculta também ser uma potência de 2. Valores comuns para o tamanho do filtro são 3 ou 5, Em geral filtros pequenos tendem a obter melhores resultados e tendem a criar redes mais profundas, e portanto serão mais poderosas.

As operações de convolução são intercaladas com operações de *pooling* e funções

de ativação, podendo ser ReLU ou outra. Em uma CNN a função de ativação não é diferente de uma RN tradicional. Para cada valor do volume,  $L_qxB_qxd_q$ , em uma camada é aplicada a função de ativação para criar valores limiares de  $L_qxB_qxd_q$ . A função ReLU não altera as dimensões de uma camada pois é um simples mapeamento um-para-um dos valores de ativação.

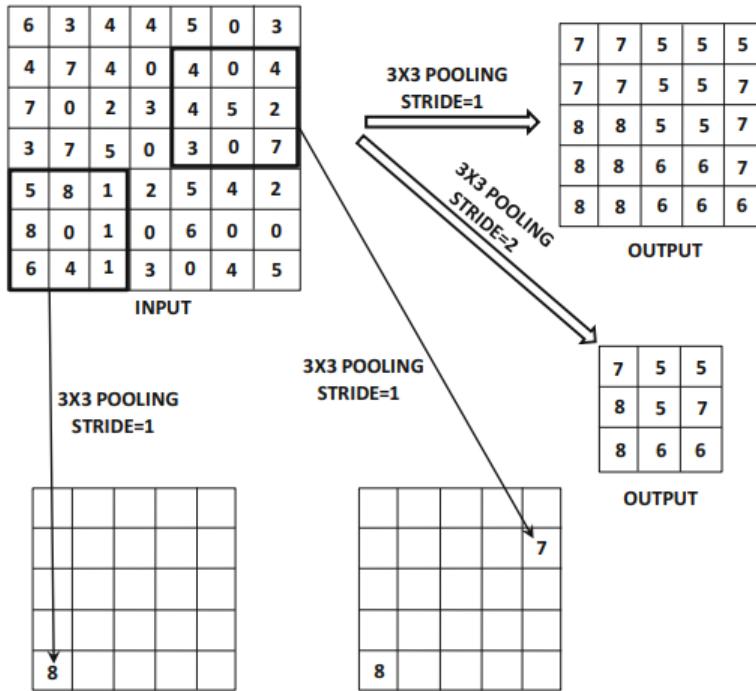
## 2.4.2 Pooling

A operação de *pooling* consiste em analisar pequenas porções da grade de tamanho  $P_qxP_q$  e produzir outra camada de mesma profundidade. Para cada região quadrada de tamanho  $P_qxP_q$  em cada um dos mapas de ativação  $d_q$ , o máximo destes valores é retornado, para essa abordagem dá-se o nome de *max-pooling*. Considerando uma *stride* de valor 1, então uma operação de *pooling* produzirá uma nova camada de tamanho  $(L_q - P_q + 1)x(B_q - P_q + 1)xd_q$ . Porém é comum usar uma passada maior que 1 no *pooling*. Nestes casos, o comprimento com a nova passada será  $1 + (L_q - P_q)/S_q$ , e a largura será  $1 + (B_q - P_q)/S_q$ . Consequentemente o *pooling* reduz drasticamente as dimensões espaciais de cada mapa de ativação.

A operação de *pooling* ocorre no nível de cada mapa de ativação. Considerando uma operação de convolução que utiliza todos os mapas de recursos  $dq$  em combinação com um filtro para produzir um valor de recurso. O *pooling* age independentemente em cada mapa de recurso para produzir outro, porém, não altera o seu número. O tamanho típico de  $P_q$  é  $2x2$ . Em uma passada, *stride*, de valor 2, não haveria sobreposição entre as regiões afetadas pelo *pooling*. Porém, em certos casos aconselha-se utilizar a sobreposição das regiões pois torna a abordagem menos propensa a *overfit*, pois a torna mais capaz de extrair as características que são necessárias para generalizações com dados não vistos, tendo em vista uma avaliação correta da performance do treinamento da RN.

Existem outros tipos de *pooling*, porém um dos mais populares continua sendo o *max-pooling*. As camadas de *pooling* são intercaladas com camadas convolucionais e/ou Ativação, embora a primeira ocorra com menor frequência em arquiteturas profundas. Isto ocorre pois as operações de *pooling* reduzem drasticamente o tamanho espacial do mapa de recursos então apenas algumas camadas de *pooling* serão necessárias para reduzir o mapa espacial a um tamanho pequeno constante. Para reduzir a *footprint* dos mapas de ativação usa-se um a operação de *pooling* de dimensões  $2x2$ . Isto resulta em uma invariância à tradução, tendo em vista que deslocar ligeiramente a imagem não altera

Figura 8 – Um exemplo de um *max pooling*.



Fonte: (LIU et al., 2018).

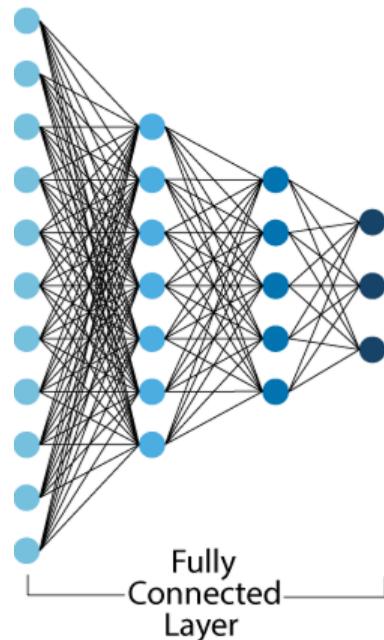
de maneira significante o mapa de ativação. Em função das imagens poderem surgir em qualquer região da imagem, torna-se fundamental a invariância à translação ajudar na classificação de maneira semelhante, quando objetos podem aparecer em diferentes regiões da imagem.

Uma nova tendência aborda a utilização de passadas maiores dentro das operações de convolução com o intuito de aumentar o campo receptivo diminuindo a pegada espacial, tornando assim desnecessária a utilização da camada de *pooling*, intercalando somente camadas convolucionais com camadas ReLU. Entretanto, esta tendência não está totalmente validada e existem argumentos a cerca de que o *max-pooling* introduz não-linearidade, algo que pode ser atingido com uma camada ReLU, e maior invariância à tradução, em comparação com convoluções ampliadas. Dessa forma, os efeitos do *pooling* não podem ser replicados por convoluções ampliadas, tornando estas operações não totalmente intercambiáveis.

Observa-se na figura 9 uma rede com camadas totalmente conectadas que fazem parte crucial de uma RNC, pois a partir dos recursos extraídos das camadas convolucionais, *pooling* e funções de ativação, conseguem estabelecer qual *label* para determinada entrada. Observa-se que essas redes possuem entre suas camadas, ligações

entre todos os neurônios.

Figura 9 – Um exemplo de uma rede *fullyconnected*.



Fonte: (HUSSEIN et al., 2021).

### 2.4.3 Treinamento e aprendizado

O aprendizado pode ser classificados em três tipos:

- Supervisionado: caracteriza-se pela existência de um supervisor externo a rede com a função de fornecer as respostas  $y_i$  para cada vetor de entrada  $X_i$ . O ajuste dos pesos é feito de maneira que a resposta  $y_i$  da rede para o vetor de entrada  $X_i$  se aproxime de  $y_i^d$  dentro dos limites preestabelecidos. As respostas da rede são comparadas as respostas do supervisor com o intuito de encontrar a direção do reajuste dos pesos.
- Não-supervisionado: caracteriza-se por não existir um supervisor, portanto não existem respostas para o treinamento, somente os dados de entradas são usados no ajuste dos pesos  $W$ . A categorização de dados é um dos problemas resolvidos por esse tipo de aprendizado.
- Reforço: Caracteriza-se por ser um paradigma intermediário aos anteriores. Utiliza-se apenas o conjunto de entrada no treinamento porém existe um crítico externo. Não existindo valores desejados na saída ao contrário de fornecer o erro

da saída da rede, retorna um sinal de reforço ou de penalização associado à última ação da rede.

O treinamento das redes neurais ocorre através da correção dos erros, em um perceptron a atualização dos pesos é obtida diretamente em direção contrária ao gradiente do erro. Em redes de múltiplas camadas são necessárias sucessivas aplicações da regra da cadeia para obter os ajustes das camadas intermediárias. A seguir observa-se na equação 2 a obtenção do ajuste dos pesos:

$$\Delta W_{ij}(n) = -\eta \frac{dE}{dW_{ij}} \quad (2)$$

O termo *retropropagation* baseia-se na retro-propagação ou também regra delta generalizada, em RNA de múltiplas camadas, dos erros para ajustes dos pesos das camadas intermediárias. Em uma rede de uma única camada as superfícies do erro são simples, podendo ser até mesmo quadráticas com um único mínimo global. Na fase inicial, a entrada é propagada até a saída, sabendo-se o valor da saída desejada  $y_i^d$  para a entrada  $x_i$  pode-se calcular o erro e através desse o ajuste dos pesos. Como não existem valores de saída para as camadas intermediárias, estas devem ser atualizadas através da retro-propagação do erro da camada de saída. Em uma rede de duas camadas, a equação de ajuste dos pesos para um neurônio da camada de saída, após a aplicação do gradiente, mostra sua forma na equação 3:

$$W_{ji}(n+1) = W_{ji}(n) + \eta e_j(n) f'_j(u_j(n)) x_i(n) \quad (3)$$

Sendo:  $f'_j(u_j(n))$  é a derivada da função de ativação do neurônio  $j$  em relação a sua saída linear  $u_j(n)$  e  $x_i(n)$  é a saída do neurônio  $i$  da camada escondida e  $\eta$  é a taxa de aprendizado. A equação 4 refere-se ao ajuste dos pesos da camada intermediária, para um melhor entendimento a observação do esquemático da equação 4 é fundamental.

$$W_{ji}(n+1) = W_{ji}(n) + \eta f'_j(u_j(n)) \sum_l e_l(n) f'_l(u_l(n)) W_{ji}(l) x_i(n) \quad (4)$$

Uma das principais dificuldades no treinamento, em algoritmos baseados em gradientes descendentes como o *backpropagation*, é a formação de regiões de gradiente nulo devido a forma irregular da superfície do erro. (REZENDE, 2003). Para evitar essas formações é o acréscimo do termo *momentum* às equações dos ajustes de peso conforme a equação 5.

$$\delta W_{ij}(n) = -\eta \frac{dE}{dW_{ij}} + \alpha \delta W_{ij}(n-1) \quad (5)$$

Em um perceptron, o processo de treino é muito simples, pois o erro pode ser calculado através de uma função direta dos pesos, o que permite uma fácil computação do gradiente. Em uma rede neural de múltiplas camadas o erro é calculado com uma complicada composição das funções dos pesos das camadas anteriores. Para calcular o gradiente, utiliza-se o algoritmo de retro-propagação, *backpropagation*. O *backpropagation* utiliza a regra da cadeia do cálculo diferencial, computando o erro do gradiente em termos de soma dos produtos do gradiente local ao longo de vários caminhos de um nó para a saída. Embora este somatório tenha um número exponencial de componentes ou caminhos, pode-se usar programação dinâmica para resultados eficientes. Existem duas fases, para frente, *forward*, e para trás, *backward*. No avanço calcula-se os valores de saída e as derivadas locais nos vários nós. E no retorno acumula-se os produtos desses valores locais em todos os caminhos do nó para a saída.

Na fase de avanço temos a alimentação da rede neural que gera uma cascata de cálculos entre as camadas, usando o conjunto atual de pesos. Então, a saída prevista é comparada com à instância de treinamento e a derivada da função de perda é computada. A derivada da perda deve agora ser calculada em relação aos pesos em todas as camadas durante a fase reversa. Na fase de retorno deve-se aprender o gradiente da função de perda em relação aos diferentes pesos. Esses gradientes são usados para atualizar os pesos em cada camada.

Considerando uma sequência de camadas escondidas  $h_1, h_2, \dots, h_k$  seguidas por uma saída  $o$  em relação a qual a função de perda é calculada. Supondo que o peso de uma camada oculta seja  $W(h_r, h_{(r+1)})$  e no caso existir somente um caminho de  $h_1$  para a saída  $o$ . Pode-se derivar o gradiente da função de perda em relação a qualquer um dos pesos de borda usando a seguinte regra da cadeia:

$$\frac{\partial L}{\partial \omega(h_{r-1}, h_r)} = \frac{\partial L}{\partial o} \left[ \frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i-1}}{\partial h_i} \right] \frac{\partial h_r}{\partial \omega_{h_{r-1}, h_r}} \quad \forall r \in 1 \dots \kappa \quad (6)$$

A expressão dada pela equação 6 considera apenas um caminho, porém pode existir uma enorme quantidade de caminhos em uma rede neural. Na equação 7 observa-se uma variante generalizada conhecida como regra da cadeia multivariável que calcula o gradiente em um gráfico computacional que pode conter outros caminhos. A expressão generalizada para um conjunto  $P$  de caminhos existente de  $h_r$  até a saída  $o$ :

$$\frac{\partial L}{\partial \omega(h_{r-1}, h_r)} = \frac{\partial L}{\partial o} \cdot \sum_{[h_r, h_{r+1}, \dots, h_k, o] \in P} \frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i-1}}{\partial h_i} \frac{\partial h_r}{\partial \omega(h_{r-1}, h_r)} \quad (7)$$

Observa-se que a partir do treinamento existem resultados que medem a performance do modelo treinado. Entre as métricas estão *precision*, *recall* e *mean average precision*. Sendo a acurácia dada pela equação 8.

$$Acuracia = \frac{TruePositives + TrueNegatives}{TruePositives + TrueNegatives + FalsePositives + FalseNegatives} \quad (8)$$

O F1-score é definido pela equação 9:

$$F1 = 2 * \frac{precision * recall}{precision + recall} \quad (9)$$

A métrica de performance *precision* é definida pela equação 10 onde temos os termos VP (Verdadeiro Positivo), VN (Verdadeiro Negativo) e FN (Falso Negativo).

$$Precision = \frac{VP}{VP + FP} \quad (10)$$

A métrica de performance *recall* é definida pela equação 11:

$$Recall = \frac{VP}{VP + FN} \quad (11)$$

#### 2.4.4 YOLOv5

O YOLOv5<sup>9</sup> é uma arquitetura capaz de detectar objetos em um único estágio, ele possui três partes importantes como qualquer outro detector de objetos de único estágio, que seriam o Modelo *Backbone*, Modelo *Neck* e Modelo *Head*. O modelo *Backbone* é principalmente usado para extrair características da imagem de entrada. No YOLOv5 foi usado CSPNet (*Cross Stage Partial Networks*) como *backbone*. CSPNet têm mostrado uma melhora no tempo de processamento através de redes mais profundas.

O CSPNet buscar permitir que as arquiteturas alcancem uma combinação de gradiente mais rica enquanto reduz a quantidade de computação. Para atender este objetivo, é partionado o mapa de recursos da camada base em duas partes e, em seguida,

---

<sup>9</sup><https://github.com/ultralytics/yolov5>

fundindo-as através de uma hierarquia entre estágios. Tendo como conceito principal fazer com que o fluxo do gradiente se propague por diferentes caminhos da rede dividindo o fluxo do gradiente. Assim confirma-se que a informação do gradiente propagado pode ter uma grande diferença de correlação, alternando etapas de concatenação e transição. Além disso o CSPNet pode reduzir a quantidade de computação e melhorar a velocidade de inferência bem como a precisão.

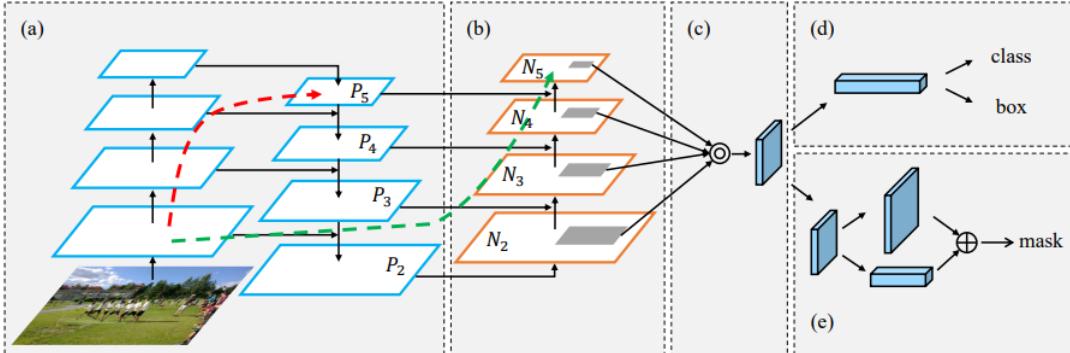
Model Neck é principalmente usado para gerar pirâmides de recursos, estes recursos ajudam os modelos em uma boa generalização no dimensionamento de objetos. As pirâmides de recursos são muito úteis ajudando os modelos a terem um bom desempenho em dados não vistos. Existem outros modelos que usam diferentes tipos de técnicas de pirâmide de recursos, como FPN (*Feature Pyramid Network*), BiFPN (*Weighted Bi-directional Feature Pyramid Network*) e PANet (*Path Aggregation Network*). O YOLOv5 utiliza o PANet para obter as pirâmides de recursos.

*Path Aggregation Network*, ou PANet, visa aumentar o fluxo de informações em uma estrutura de segmentação de instância baseada em proposta. Especificamente, a hierarquia de recursos é aprimorada com sinais de localização precisos em camadas inferiores por aumento de caminho de baixo para cima, que encurta o caminho de informações entre as camadas inferiores e o recurso superior. Além disso, o *pooling* de recursos adaptáveis é empregado, que vincula a grade de recursos e todos os níveis de recursos para fazer com que informações úteis em cada nível de recurso se propaguem diretamente para as seguintes sub-redes de propostas. Um ramo complementar que captura diferentes visualizações para cada proposta é criado para melhorar ainda mais a previsão da máscara.(LIU et al., 2018).

Pode-se observar a figura 10 que ilustra a PANet, composta por diferentes partes, dividida em (a) *FPN backbone*, (b) *Bottom-up, path augmentation*, (c) *Adaptive feature pooling*, (d) *Box branch* e (e) *Fully-connected fusion*.

O modelo Head é usado principalmente para realizar a parte final da detecção. Ele aplica caixas de âncora em recursos e gera vetores de saída final com as probabilidades de classe, pontuação de objetividade e caixas limitadoras. A escolha da função de ativação é crucial em qualquer rede neural profunda, recentemente foram introduzidas diversas funções de ativação, como Leaky ReLU, mish, swish, entre outras. Os autores decidiram utilizar a LeakyReLU e a função sigmoid. sendo a LeakyReLU utilizada nas camadas intermediárias ou escondidas e a função sigmoid na camada de detecção final.(XU et al., 2015) (MISRA, 2019) (CELEBI; CEYLAN, 2019)

Figura 10 – Ilustração PANet e seus componentes.



Fonte: (LIU et al., 2018).

No YOLOv5 temos duas opções como função de otimização, primeiramente temos *Stochastic Gradient Descent*, SGD , que se não alterado em código, será o utilizado. Assim pode-se mudar para a função de otimização Adam. Adam combina as melhores propriedades dos algoritmos AdaGrad e RMSProp para fornecer um algoritmo de otimização que pode lidar com gradientes esparsos, (ZOU et al., 2019).

Na imagem 11 observa-se o comparativo entre a performance dos modelos de YOLOv5, diferenciando-se por tamanho e número de camadas e parâmetros.

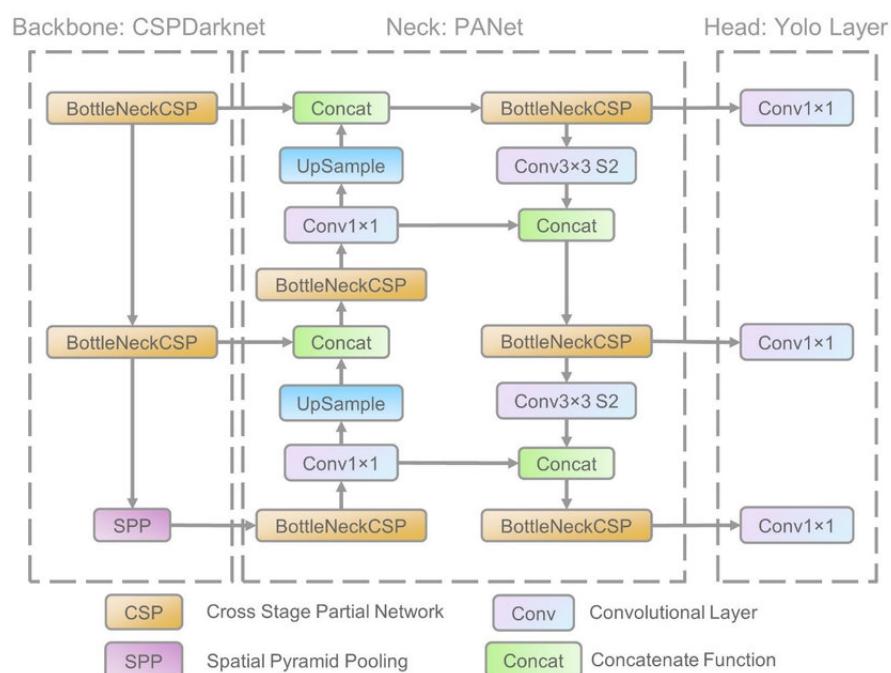
Figura 11 – Desempenho dos diferentes modelos do YOLOv5.

Model	size (pixels)	mAP <sub>val</sub> 0.5:0.95	mAP <sub>val</sub> 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	<b>45</b>	<b>6.3</b>	<b>0.6</b>	<b>1.9</b>	<b>4.5</b>
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

Fonte: [github.com/ultralytics/yolov5](https://github.com/ultralytics/yolov5)

O YOLOv5 possui em seu *backbone* o CSPDarknet com 19 camadas convolucionais e 5 *max pooling*. Observa-se na imagem 12 o modelo *backbone*, representado pela CSPDarknet, que é a incorporação do CSP na rede Darknet. O modelo *neck* busca otimizar o fluxo de informações através de sua estrutura de pirâmide melhorando o fluxo ascendente dos dados

Figura 12 – Ilustração da arquitetura do YOLOv5.



Fonte: (XU et al., 2021)

### 3 TRABALHOS CORRELATOS

Tendo em vista o ótimo desempenho dos detectores de objetos em suas tarefas, esse tipo de rede neural ganhou muito destaque no cenário de inteligência artificial. Observa-se que diversos trabalhos que utilizam as redes neurais convolucionais na detecção de objetos ou pessoas, levando em consideração que esta é a técnica mais efetiva para tal tarefa. Atualmente a maioria dos trabalhos que buscam a detecção de armas giram em torno das redes neurais convolucionais, configurando a arquitetura e os hiper-parâmetros de diferentes maneiras, utilizando diferentes *dataset* no seu treinamento e também integrando com outras IA na busca de criar um sistema mais robusto, que seja capaz de minimizar os erros e maximizar os acertos.

#### **3.1 Modelos convolucionais na detecção de armas.**

O artigo de Romero e Salamea (2019) com título traduzido de "Modelos Convolucionais para Detecção de Armas de Fogo em Vídeos de Vigilância", destacando que desejavam apenas realizar a buscar por armas em áreas da imagem que tinham pessoas evitando áreas irrelevantes. O artigo aborda que foram utilizadas diversas arquiteturas de CNN encontrando valores acima de 86% de *recall* e precisão usando uma rede VGG, também conhecida por OxfordNet, em imagens em *grayscale*. O *database* foi construído com imagens e vídeos retirados do Instagram<sup>10</sup>, Youtube<sup>11</sup> e Google<sup>12</sup>. Foram utilizadas no total 17.684 imagens divididas em duas classes: com arma e sem arma. A estrutura de cada classe é mostrada na figura 13.

São utilizadas imagens de diversas origens, utilizando imagens de vídeos de segurança de roubos e pessoas praticando tiro, sendo estas imagens escolhidas pois se assemelham ao encontrado em situações reais. O modelo deve conhecer imagens de baixa qualidade e luminosidade. Então o *database* deve conter imagens de diferentes qualidades e posições como mostra a figura 14 e a figura 15.

Inicialmente foram usados 70% das imagens para treino 15% para validação e 15% para testes. O sistema de detecção foi desenvolvido usando Python<sup>13</sup> e Tensorflow

---

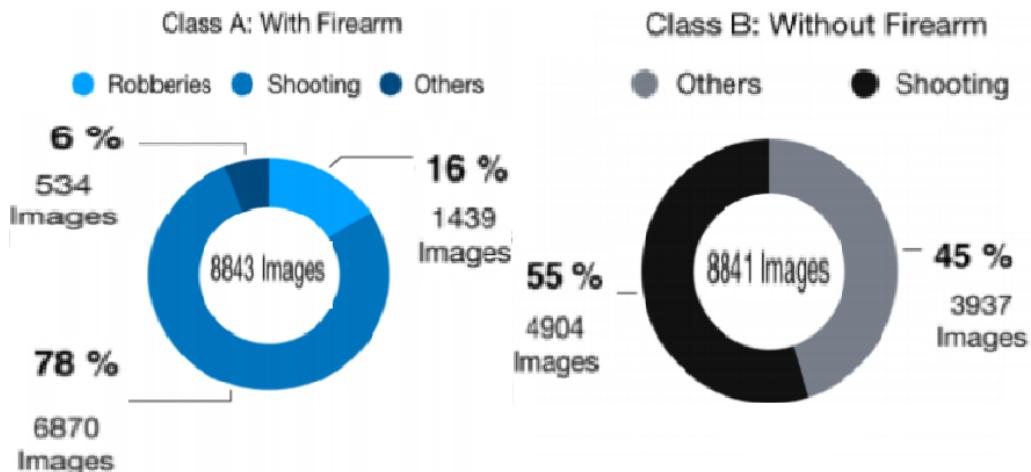
<sup>10</sup>[www.instagram.com](http://www.instagram.com)

<sup>11</sup>[www.youtube.com](http://www.youtube.com)

<sup>12</sup>[www.google.com](http://www.google.com)

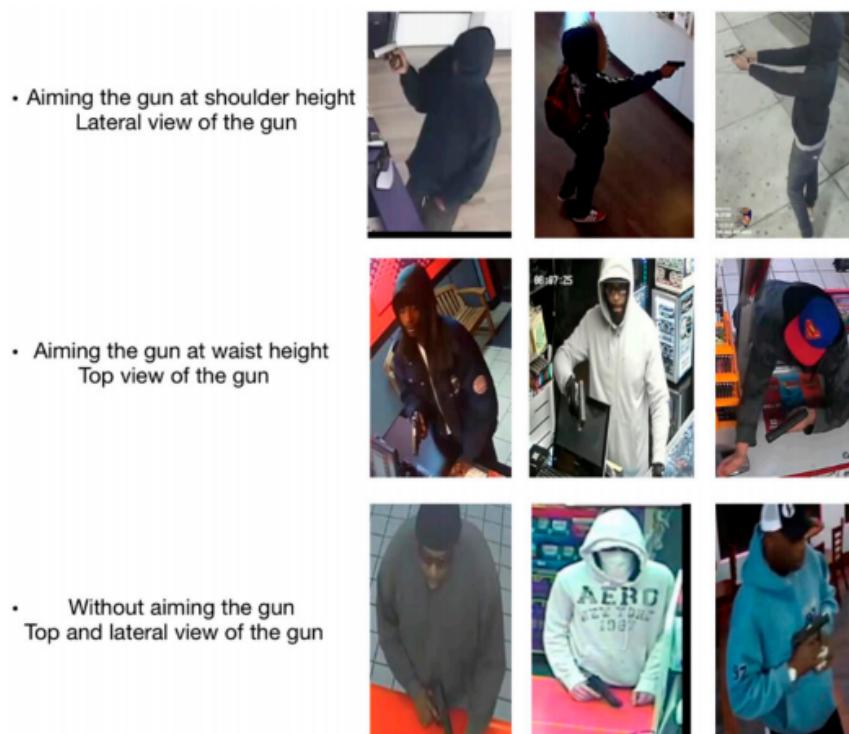
<sup>13</sup><http://www.python.org>

Figura 13 – Imagem da classe A na esquerda e da classe B na direita.



Fonte: Romero e Salamea (2019).

Figura 14 – Diferentes posições com a arma.



Fonte: Romero e Salamea (2019).

<sup>14</sup>. Tendo em vista a necessidade de ampliar o número de imagens, foram utilizadas múltiplas técnicas como girar a imagem no eixo horizontal como também rodar em diversos ângulos. Através dessas técnicas ampliou-se o *database* de 17.684 para 247.576.

<sup>14</sup><https://www.tensorflow.org>

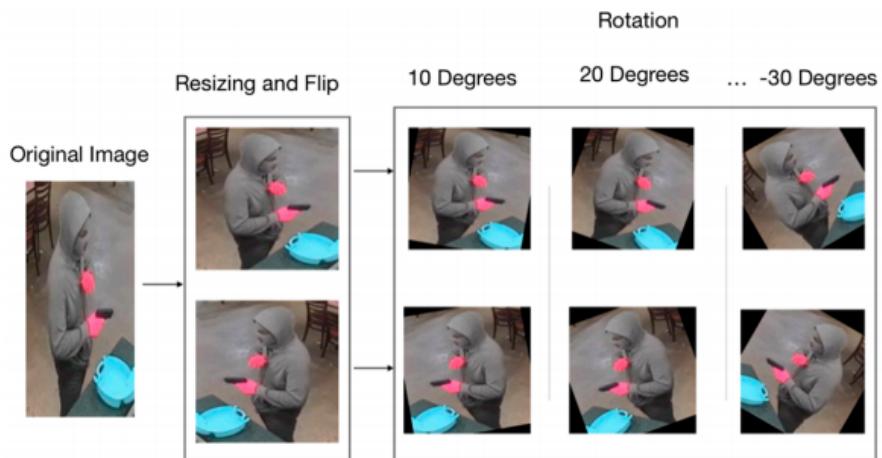
Figura 15 – Diferentes intensidades de luz.



Fonte: Romero e Salamea (2019).

A figura 16 demonstra as técnicas usadas.

Figura 16 – Técnicas de aumento do *database*

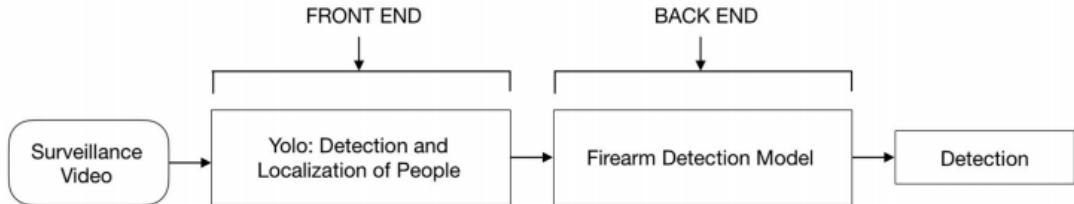


Fonte: Romero e Salamea (2019).

A arquitetura do sistema desenvolvido é formada por um *pipeline* de RN treinadas para tarefas diferentes. Iniciando com um *front-end* composto pelo YOLO, com um treinamento no *database* chamado "COCO" que contém 15 classes como: pessoa, cachorro, gato, bicicleta, avião, carro, ônibus, trem, barco, pássaro, vaca, elefante, ovelha, mochila, gravata, faca, colher, banana, maçã, laranja, sanduíche, entre outros. O intuito desse primeiro estágio é primeiramente buscar pessoas nas imagens e então enviá-las para o segundo estágio. Como *back-end* temos o modelo de detecção de pessoas armadas com o mesmo objetivo de desenvolvido desse trabalho. O sistema usa como entrada um vídeo de vigilância e como saída a detecção de uma pessoa armada ou não. A figura 17 mostra

a arquitetura.

Figura 17 – Arquitetura do sistema



Fonte: Romero e Salamea (2019).

A figura 18 mostra um exemplo da segmentação e descarte, onde pode-se notar a área recortada através da moldura vermelha ao redor da pessoa armada e o restante da imagem em volta que será descartada para análise posterior feita pelo segundo estágio que é treinado para identificar as armas, visando uma melhoria no desempenho, tendo em vista uma área menor de busca, sendo as armas somente procuradas pelo segundo estágio.

Figura 18 – Segmentação e descarte de áreas desnecessárias.



Fonte: Romero e Salamea (2019).

O projeto obteve o melhor desempenho utilizando uma arquitetura de rede VGG, obtendo uma melhora de 21,4% na perda e 3,33% na precisão, em comparação com uma arquitetura de rede ZF Net. O uso de imagens em tons de cinza melhorou em 0,22% a

precisão e 34,3% na perda em comparação as imagens RGB. Na execução final o sistema obteve 86% de precisão. Podemos observar as configurações das redes, A figura 19 mostra a arquitetura VGG e suas camadas de convolução seguidas de *Max-Pooling* finalizando com uma camada SoftMax, vale ressaltar a existência de duas configurações na imagem, diferentes apenas na camada da função SoftMax. Na figura 20 é abordado a arquitetura ZF Net.

Figura 19 – Arquitetura VGG Net.

<b>Configurations</b>	
<b>C1</b>	<b>C2</b>
Conv $3 \times 3 - 64$	Conv $3 \times 3 - 64$
Max-Pooling $2 \times 2$	
Conv $3 \times 3 - 128$	Conv $3 \times 3 - 128$
Max-Pooling $2 \times 2$	
Conv $3 \times 3 - 256$	Conv $3 \times 3 - 256$
Max-Pooling $2 \times 2$	
Conv $3 \times 3 - 512$	Conv $3 \times 3 - 512$
Max-Pooling $2 \times 2$	
Conv $3 \times 3 - 512$	Conv $3 \times 3 - 512$
Max-Pooling $2 \times 2$	
FC - 2048	FC - 4096
FC - 2048	FC - 2
	SoftMax

Fonte: Romero e Salamea (2019).

### 3.2 Comparativo de diversas arquiteturas e técnicas.

Neste artigo de Bhatti et al. (2021) é abordado a utilização de diversas RN com o objetivo de identificar armas em imagens de CCTV (*Closed Circuit Television*). Foram utilizadas as técnicas de *sliding window/classification* (classificadores) e *region proposal/object detection* (detectores de objetos). Alguns dos algoritmos usados são VGG16 (HASSAN, 2018), Inception-V3 (SZEGEDY et al., 2016), Inception-ResnetV2 (SZEGEDY et al., 2017), SSDMobileNetV1 (LIU et al., 2016), Faster-RCNN InceptionResnetV2 (FRIRV2) (REN et al., 2015), YOLOv3 (REDMON; FARHADI, 2018) e YOLOv4 (BOCHKOVSKIY; WANG; LIAO, 2020). *Precision and recall* valem mais que *accuracy* como parâmetro para este estudo.

Figura 20 – Arquitetura ZF Net.

Configurations		
C3	C4	C5
Conv $7 \times 7 - 64$	Conv $7 \times 7 - 92$ Max-Pooling $3 \times 3$	Conv $7 \times 7 - 92$
Conv $5 \times 5 - 128$	Conv $5 \times 5 - 192$ Max-Pooling $3 \times 3$	Conv $5 \times 5 - 192$
Conv $3 \times 3 - 192$ Conv $3 \times 3 - 192$ Conv $3 \times 3 - 128$	Conv $3 \times 3 - 256$ Conv $3 \times 3 - 256$ Conv $3 \times 3 - 192$ Max-Pooling $3 \times 3$	Conv $3 \times 3 - 256$ Conv $3 \times 3 - 256$ Conv $3 \times 3 - 192$ Conv $3 \times 3 - 192$
FC - 2048 FC - 2048 FC - 2	FC - 4096 FC - 2 SoftMax	FC - 4096 FC - 2048 FC - 2
	SoftMax	

Fonte: Romero e Salamea (2019).

O YOLOv4 obteve a melhor performance atingindo um *F1-score* de 91% e um *mean average precision* de 91,73% sendo este número superior ao alcançado previamente. O *dataset* foi construído com imagens da internet, vídeos do youtube, da Universidade de Granada, repositórios do github e do site "imfdb.org". As classes utilizadas foram pistola e não pistola, tendo pistola, revólver e outras armas como categorias da classe pistola e carteira, *selfie stick* telefone e detector de metais como categorias da classe "não pistola", que existe com o objetivo de diminuir o número de falso positivos tendo em vista a grande possibilidade da confusão destes itens com armas. A figura 21 mostra imagens da classe não pistola.

Foram utilizados 3 *dataset*, sendo o *dataset 1* utilizado no inicio dos trabalhos contendo no total 1732 imagens divididas em 750 na classe pistola e 950 na classe "não pistola", o *dataset 2* foi construído para um cenário *real-time*, classificadores, e contém 5254 imagens e o *dataset 3* foi construído para o cenário *real-time* e os algoritmos de detecção de objetos, partiu do *dataset 1* e através da adição de mais imagens e *data augmentation* passou a possuir 8327 imagens dividida nas duas classes. A divisão das imagens para treinamento e testes foi de 12% no *dataset 3* e 15% no *dataset 1* e *dataset 2*.

A performance destes modelos foi analisada comparando-as em termos da

Figura 21 – Imagens da classe não pistola.



Fonte: Bhatti et al. (2021)

*standart metrics of F1-score e mean average precision (mAP) para melhor performance. Estes termos foram calculados a partir das equações 12, 13 e 14 . F1 score é uma taxa entre precision e recall functions. O dataset 1 foi utilizado pelos classificadores e seus resultados estão na tabela 1.*

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (12)$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (13)$$

$$\text{F1-score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (14)$$

<b>Algorithms</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>
VGG16	71%	66,66%	69,09%
Inceptionv3	74,11%	96,18%	83,71%
InceptionResNetV2	79,24%	89,54%	84,07%

Tabela 1 – Resultado dos classificadores no *dataset1*.

Após esta experimentação, analisando e definindo que não foram bons os resultados, a equipe concluiu que as imagens do *dataset* majoritariamente possuíam um fundo branco ou um mesmo tipo de fundo, o que levava ao modelo considerar o fundo

como uma ROI (*Region Of Interest*). Então foi desenvolvido um novo *dataset* com imagens em diversos casos e fundos, que seria o *dataset 2*. Na tabela 2 observa-se os resultados do treinamento com o *dataset 2*.

<i>Algorithms</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>
VGG16	80%	83,47%	81,69%
Inceptionv3	84,36%	84,36%	84,36%
InceptionResNetV2	85,52%	85,98%	85,74%

Tabela 2 – Resultado dos classificadores no *dataset 2*.

O *dataset 3* foi utilizado para treinar os algoritmos detectores de objetos. Cada um desses algoritmos tem seus prós e contras, SSDMobileNet é muito bom em FPS. FasterRCNN-InceptionResNetv2 é boa em *precision* e *recall* porém ruim em velocidade de processamento. Diferente de outros métodos de regiões de proposta, a série YOLO divide a imagem em uma grade *SxS* e simultaneamente prevê *boundingbox* para o objeto e sua probabilidade. A tabela 3 mostra os resultados dos detectores de objetos treinados no *dataset 3*.

<i>Models</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>
SSDMobileNetV1	62,79%	60,23%	59%
YOLOv3	85,86%	87,34%	86%
Faster-RCNN InceptionResnetV2	86,38%	89,25%	87%
YOLOv4	93%	88%	91%

Tabela 3 – Resultado dos detectores de objetos no *dataset 3*, IoU *threshold* de 50%.

### 3.3 IoT Based Weapons Detection System

No trabalho de (SINGH et al., 2021) observa-se que com o avanço da IoT (*Internet of Things*) certamente veremos RN funcionando em dispositivos embarcados como: Arduino Uno Rev3<sup>15</sup>, ESP32<sup>16</sup>, Nvidia Jetson<sup>17</sup> e Raspberry Pi 4<sup>18</sup>. Este estudo visa destacar as diferenças de performance alcançadas com o YOLOv4 utilizando diferentes opções de *hardware*. Destaca-se que a utilização de dispositivos para IoT conta com a vantagem da comunicação sem fio e a disposição de uma série de outros sensores capazes de aumentar a consciência situacional.

<sup>15</sup><https://www.arduino.cc>

<sup>16</sup><https://www.espressif.com>

<sup>17</sup><https://www.nvidia.com/pt-br/autonomous-machines/embedded-systems/>

<sup>18</sup><https://www.raspberrypi.org>

O presente projeto aborda a utilização de projetos de detecção de armas em aeroportos, estações de trem, terminais de ônibus, bancos, bases militares e casas. Vale ressaltar a utilização do sistema de uma maneira estática com CCTV posicionado no alto de uma construção ou esquina, e dinâmica, com o sistema funcionando em um UAV (*Unmanned Aerial Vehicle*) ou UGV (*Unmanned Ground Vehicle*).

Seus resultados demonstram que o desempenho vai depender do hardware utilizado e a qualidade da imagem. Passando de 70% em imagem de baixa qualidade para 95% em imagem de alta qualidade. Usou-se imagens do Kaggle e Google na construção do *dataset*, no treinamento foram adotadas 3000 iterações. Apesar do *dataset* conter imagens de faca, espada, pistola, metralhadora e escopeta, foi utilizado apenas uma classe chamada arma.

Nesse projeto foi necessário o ajuste de diversos parâmetros para o treinamento, como: *batch*, *subdivisions*, *width*, *height*, *channels*, *momentum*, *decay*, *angle*, *saturation*, *exposure*, *hue*, *learning rate*, *burn in*, *max batches*, *policy*, *steps* e *scales*, e para os testes, parâmetros como: *batch* e *subdivision*, todavia ao utilizar o YOLOv5, para alguns desses parâmetros não são necessários esses ajustes, sendo eles automáticos, como a definição do tamanho da imagem de entrada.

Observa-se na tabela 4 o comparativo entre este trabalho e os correlatos. Destaca-se que mesmo não utilizando técnicas de *data augmentation*, o modelo não será prejudicado no treino, pois o YOLOv5 possui esta técnica internamente. Além de observar a qualidade das imagens tendo em vista o controle sobre a criação delas. Destacar a criação do *dataset* próprio que é a base para o treino.

<b>Trabalho</b>	<b>Dataset próprio</b>	<b>DataAugmentation</b>	<b>Série YOLO</b>
Romero e Salamea (2019)		X	
Bhatti et al. (2021)		X	X
SINGH et al. (2021)		X	X
Autor próprio	X	X	X

Tabela 4 – Comparativo de técnicas e *framework* usados.

## 4 CRIAÇÃO DO DATASET E TREINAMENTO

Nesta seção são abordados os procedimentos de desenvolvimento do projeto. A presente proposta consiste na utilização o YOLOv5 em um sistema composto por uma RN treinada para identificar armas que ao encontrar alguma arma, como: pistola ou revólver. Na figura 22 observa-se o funcionamento da RN treinada para identificar armas, contendo na mesma uma pessoa mirando uma arma em outra. Nota-se que existe ao redor da arma uma *bound box* que a delimita, uma numeração que quantifica a confiabilidade da detecção e o nome de sua classe.

Figura 22 – Exemplo de identificação de arma pela RN.



Fonte: Autor

Nesta parte é abordada a plataforma utilizada para o desenvolvimento do projeto experimental e os motivos de sua escolha. A escolha da plataforma *Google Colab*<sup>19</sup> foi devido sua facilidade de operação que permite codificação no próprio navegador além de permitir a utilização das placas gráficas do próprio *Google*, tendo em vista o gasto energético e tempo de treinamento necessário ao desenvolvimento do projeto. Porém, a

<sup>19</sup><https://colab.research.google.com/notebooks/welcome.ipynb?hl=pt-BR>

plataforma do google colab no modo gratuito, possui limitações, tais como: a performance da GPU disponibilizada, o tempo de utilização. Devido a isso faz-se necessário o particionar o treinamento com o intuito de respeitar essas limitações impostas.

O *dataset* é uma parte fundamental no projeto pois se trata de uma RN com treinamento supervisionado. Entretanto, para se atingir um nível satisfatório de desempenho deve-se utilizar um *dataset* vasto, com imagens com diferentes fundos e situações.

A validação dos resultados das redes neurais ocorre através de um *dataset* de validação. Essas imagens do *dataset* são desconhecidas do treinamento e portanto através dessas novas imagens que são avaliados os parâmetros como mAP, GIOU, precisão, acurácia e outros.

#### **4.1 Dataset Inicial**

Inicialmente foi desenvolvido um *dataset* utilizando somente imagens do site imfdb.org, cujo conteúdo apresenta vasto material gráfico sobre armas em suas diversas variações contendo aproximadamente 1475 imagens. Utilizou-se quatro classes, compostas por revólver, pistola, escopeta e fuzil.

- A classe pistola contém 420 imagens.
- A classe revólver contém 241 imagens.
- A classe escopeta contém 140 imagens.
- A classe fuzil é contém 578 imagens.

Na figura 23 observa-se que a arma compõe quase 100% do total da imagem, diferente do modo que se apresentará em uma CCTV (Closed-circuit Television) ou qualquer outra imagem com um contexto, como uma pessoa portando-a e um fundo que adicionam características à imagem alterando como será observada.

Figura 23 – Exemplo de imagem de fuzil modelo AR do *database* do site *Internet Movie Firearms database*.



Fonte: imfdb.org

Na figura 24 observa-se que existem quatro militares armados porém a RN detecta como uma única arma usando apenas uma *bound box*, diferente do resultado desejado que seriam três fuzis e uma escopeta. Este resultado é oriundo da forma como o *dataset* foi confeccionado, utilizando imagens somente de armas sem qualquer contexto, como pessoas portando-as ou fundo. O erro de utilizar apenas uma única e grande *bound box* é oriundo da arma, no *dataset*, compor quase 100% da imagem, confundindo a RN no momento do treinamento e na detecção.

Figura 24 – Exemplo de imagem analisada pela RN treinada com o *dataset* inicial.



Fonte: Autor

Observa-se os resultados adquiridos com o treinamento da RN com o *dataset* inicial. Portanto, conclui-se que é necessário utilizar imagens semelhantes com as utilizadas na detecção, possuindo uma pessoa portando a arma e um fundo. O *dataset* de imagens de armas sem um contexto e um ambiente, fizeram com que esse primeiro teste não obtivesse um bom resultado. Além disso, destaca-se que na detecção, tendo em vista a baixa qualidade da imagem da arma, não possui muitos detalhes visuais. Portanto, a detecção se dará pela utilização do conjunto da arma com o braço e mão do portador.

#### 4.2 Dataset Próprio

Ao analisar os resultados do *dataset* inicial, conclui-se que é necessária a criação de um novo, na busca pelo aprimoramento da detecção utilizando imagens de ângulos e situações semelhantes às encontradas no objetivo fim do projeto. Este *dataset* próprio é composto somente por imagens de revólver e pistola. Possuindo 2934 imagens com pessoas portando armas em suas mais diversas posições, com diferentes fundos e roupas.

Destaca-se que a detecção, por ser feita analisando os *pixels* da imagem, é influenciada tanto pelo contexto, como fundo da imagem ou roupas vestidas pelo portador da arma, quanto pela mão do portador, tendo em vista que por se tratar de um objeto pequeno, a mão o segurando torna-se uma grande parte da imagem a ser identificada.

Utilizando uma *boundingbox* que inclua a mão do operador fica suscetível a absorver as características da mão e gerar falsos positivos em imagens com outros objetos sendo portados ou somente a mão. Na figura 25 observa-se a diferença de qualidade da imagem entre uma foto e um vídeo, ambos gravados com a mesma câmera. Além disso, destaca-se a falta de detalhes do objeto juntamente com a predominância da mão do portador em relação ao objeto, problema que é agravado com a perda de qualidade gráfica devido ser um vídeo.

Figura 25 – Na imagem à esquerda observa-se uma foto com uma câmera de 48 *pixels* e à direita um recorte realizado pela RN de um vídeo.



Fonte: Autor

Tendo em vista o objeto pode se apresentar em diversas posições, sendo ele um objeto tridimensional, alterando sua forma bidimensional a qual é utilizada na detecção. Diversos ângulos de visão tornam tais objetos totalmente diferentes de sua aparência esperada, tornando-o semelhante a outros objetos, consequentemente gerando falsos

positivos.

Na figura 26 observa-se à esquerda que uma pessoa portando um controle remoto de televisão com uma pegada semelhante a uma arma, torna confusa, até mesmo para um observador humano, a identificação do objeto. Na imagem à direita, pode-se ver que um armamento pode ser facilmente confundido com um objeto do cotidiano, como um controle remoto.

Figura 26 – À esquerda pessoa portando um controle remoto e à direita portando uma pistola.



Fonte: Autor

Tendo em vista os problemas de falsos positivos e a falta de detalhes nesses objetos pequenos que são confundidos com armas, torna-se evidente a necessidade de ampliação do *dataset* e a existência de um operador humano para a confirmação da aparição do armamento na imagem. O funcionamento totalmente automatizado, sem operador humano, exige câmeras com melhor resolução e velocidade de captação de quadros, FPS (*frames per second*).

Observou-se que o objeto a ser detectado pode se apresentar em diferentes pontos de vista, variando a quantidade de detalhes que são possivelmente captados, como também a forma que o objeto terá. Portanto, existindo diferentes situações que podem facilitar ou dificultar a detecção. Na criação do *dataset* foram priorizadas imagens que contemplam as mais diversas formas do objeto se apresentar. A única opção descartada é quando o objeto está totalmente escondido na imagem pelo próprio portador dele, sendo o pior caso, como vê-se na figura 27, situação onde torna-se impossível a detecção da arma.

Figura 27 – Pessoa armada de costas.



Fonte: Autor

Destaca-se o melhor caso, de acordo com o posicionamento da arma, que seria o observador, câmera captadora, estar a 90° do portador do objeto. Obtendo assim uma imagem com uma silhueta característica e o máximo de detalhes a ser utilizado na detecção. Nota-se na figura 28 que este posicionamento favorece ao máximo a detecção, se for também levada em consideração o contexto do objeto.

Figura 28 – Pessoa armada a 90° com contexto limpo.



Fonte: Autor

Partindo da análise do pior caso, onde é impossível detectar o objeto e passando

pelo melhor caso, onde tem-se as melhores condições para a detecção. Existem infinitos ângulos de visão que alteram o ponto de vista do observador e consequentemente a estrutura como o objeto se apresenta.

Uma das posições onde o objeto tem menos detalhes, portanto, tornando-se um dos piores ângulos de visão para detecção, é quando o portador está mirando na câmera ou próximo dela, como pode-se ver na figura 29 que, diferente do melhor caso, o objeto mudou drasticamente seu formato e tem seu tamanho diminuído.

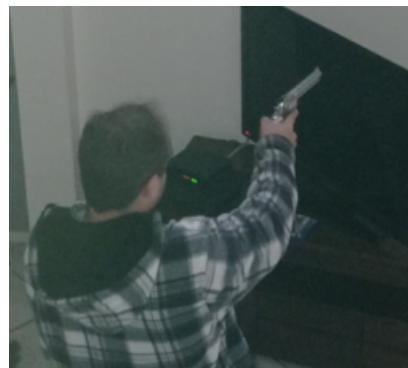
Figura 29 – Pessoa armada mirando para a câmera.



Fonte: Autor

Observa-se na figura 30 que pode-se ter uma visão de um ângulo intermediário pelas costas do portador do objeto, que dependendo da posição da câmera no ambiente é uma das mais prováveis formas de ponto de vista do observador e onde ainda existem detalhes a serem levados em consideração na detecção, porém, o objeto muda a estrutura pelo ponto de vista do observador, diminuindo seu tamanho e mudando sua forma. O que consequentemente aumenta a dificuldade da detecção, tendo em vista essa perda de detalhes e tamanho em relação ao melhor caso.

Figura 30 – Pessoa armada parcialmente de costas para câmera.



Fonte: Autor

Observa-se na figura 31 que pode-se ter também um ângulo frontal ao portador entretanto intermediário, onde muda drasticamente o formato do objeto a ser detectado, adquirindo características distintas de uma visão traseira. Além de notar-se um aumento no tamanho e detalhes da mão do portador.

Figura 31 – Pessoa armada parcialmente de frente para câmera.



Fonte: Autor

Observa-se na figura 32 que o objeto a ser detectado pode se apresentar com um contexto que dificulte sua detecção, como neste caso, pode ter sua cor semelhante ao contexto. Podendo ser desde a vestimenta do portador quanto ao fundo da imagem.

Figura 32 – Objeto com contexto de cor semelhante a ele.



Fonte: Autor

Nota-se que na figura 33 tem-se uma pessoa armada de uma posição quase frontal, porém com o armamento lateralizado, o que do ponto de vista do ângulo tem-se uma boa imagem para detecção. Porém, nesta imagem o armamento está logo a frente de uma vestimenta que acaba compondo seu contexto e atrapalha na detecção, pois acrescenta

linhas que diferem do formato original do objeto. Tais linhas confundem a RN pois divergindo do formato original que é captado no mapa de características dificulta e muito a detecção destes casos. Situação que pode ser agravada com a cor do contexto ser igual a do objeto.

Figura 33 – Objeto com contexto que gera confusão.



Fonte: Autor

Diante dos exemplos de situações já destacadas, fica evidente a necessidade de um *dataset* vasto e composto por imagens de diferentes armamentos, de diferentes formatos e cores, em diferentes ângulos de visão. Alterando o contexto da cena onde o objeto está inserido, mudando o padrão e as cores que estão logo atrás do objeto. Sendo estes a vestimenta do portador e o fundo da cena, que esteja logo atrás do objeto a ser detectado e que pode influenciar a imagem a ser captada e utilizada no processo de detecção.

Os treinamentos foram realizados sobre dois *dataset* finais, contendo imagens que abordam todos os possíveis pontos de vista do observador, posicionamento do objeto, contexto da cena e contexto da vestimenta do portador. Diferenciando-se apenas no número de imagens utilizadas. Com o menor deles contendo um total 2.934 imagens e o maior contendo 4.369 imagens, ambos sendo divididos da proporção de 80% para treino e 20% para a validação do modelo.

Foi observado que o tamanho do modelo utilizado não influenciou drasticamente

o resultado do treino, porém, ficou evidente a melhoria diante da ampliação do número de imagens, variações de armas, contexto da cena e a vestimenta do portador.

Como observa-se na figura 34 os hiperparâmetros usados nos treinamentos, ressaltando alguns como a taxa de aprendizado, que controla o quanto altera o modelo em resposta ao erro estimado a cada atualização dos pesos, exemplificados como lr0 e lrf no topo da imagem 34 e que estão em seu valor padrão.

O *momentum* que pode ser descrito como uma taxa aplicada ao gradiente descendente da iteração passada que, combinado com o gradiente da iteração atual, ajuda o modelo a convergir mais rapidamente e não ficar preso em mínimos locais.

Figura 34 – Hiperparâmetros para o treino.

```
lr0: 0.01
lrf: 0.01
momentum: 0.937
weight_decay: 0.0005
warmup_epochs: 3.0
warmup_momentum: 0.8
warmup_bias_lr: 0.1
box: 0.05
cls: 0.5
cls_pw: 1.0
obj: 1.0
obj_pw: 1.0
iou_t: 0.2
anchor_t: 4.0
fl_gamma: 0.0
hsv_h: 0.015
hsv_s: 0.7
hsv_v: 0.4
degrees: 0.0
translate: 0.1
scale: 0.5
shear: 0.0
perspective: 0.0
flipud: 0.0
fliplr: 0.5
mosaic: 1.0
mixup: 0.0
copy_paste: 0.0
```

Fonte: Colab

Na figura 35 da arquitetura *small* pode-se observar suas camadas e seus respectivos tamanhos. Claramente se diferem em tamanho e número de camadas da figura 36 que é de uma arquitetura *medium*. A rede *small* 270 camadas com 7.022.326 parâmetros, 7.022.326 gradientes e 15.9 GFLOPS, (*Giga Float Point Operations Per Second*). A rede *medium* contém 369 camadas com 20.871.318 parâmetros, 20.871.318 gradientes e 48.2 GFLOPS.

Figura 35 – Arquitetura *Small*.

```

      from n    params   module           arguments
0       -1 1     3520  models.common.Conv [3, 32, 6, 2, 2]
1       -1 1    18560  models.common.Conv [32, 64, 3, 2]
2       -1 1    18816  models.common.C3  [64, 64, 1]
3       -1 1    73984  models.common.Conv [64, 128, 3, 2]
4       -1 2   115712  models.common.C3  [128, 128, 2]
5       -1 1   295424  models.common.Conv [128, 256, 3, 2]
6       -1 3   625152  models.common.C3  [256, 256, 3]
7       -1 1   1180672 models.common.Conv [256, 512, 3, 2]
8       -1 1   1182720 models.common.C3  [512, 512, 1]
9       -1 1   656896  models.common.SPPF [512, 512, 5]
10      -1 1   131584  models.common.Conv [512, 256, 1, 1]
11      -1 1     0  torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
12      [-1, 6] 1     0  models.common.Concat [1]
13      -1 1   361984  models.common.C3  [512, 256, 1, False]
14      -1 1   33024  models.common.Conv [256, 128, 1, 1]
15      -1 1     0  torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
16      [-1, 4] 1     0  models.common.Concat [1]
17      -1 1   90880  models.common.C3  [256, 128, 1, False]
18      -1 1   147712  models.common.Conv [128, 128, 3, 2]
19      [-1, 14] 1     0  models.common.Concat [1]
20      -1 1   296448  models.common.C3  [256, 256, 1, False]
21      -1 1   590336  models.common.Conv [256, 256, 3, 2]
22      [-1, 10] 1     0  models.common.Concat [1]
23      -1 1   1182720 models.common.C3  [512, 512, 1, False]
24      [17, 20, 23] 1   16182  models.yolo.Detect [1, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119],
                                                [116, 90, 156, 198, 373, 326]], [128, 256, 512]]
```

Model summary: 270 layers, 7022326 parameters, 7022326 gradients, 15.9 GFLOPS

Fonte: Colab

Figura 36 – Arquitetura *Medium*.

```

      from n    params   module           arguments
0       -1 1     5280  models.common.Conv [3, 48, 6, 2, 2]
1       -1 1    41664  models.common.Conv [48, 96, 3, 2]
2       -1 2    65280  models.common.C3  [96, 96, 2]
3       -1 1   166272  models.common.Conv [96, 192, 3, 2]
4       -1 4   444672  models.common.C3  [192, 192, 4]
5       -1 1   664320  models.common.Conv [192, 384, 3, 2]
6       -1 6   2512896 models.common.C3  [384, 384, 6]
7       -1 1   2655744  models.common.Conv [384, 768, 3, 2]
8       -1 2   4134912 models.common.C3  [768, 768, 2]
9       -1 1   1476864  models.common.SPPF [768, 768, 5]
10      -1 1   295680  models.common.Conv [768, 384, 1, 1]
11      -1 1     0  torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
12      [-1, 6] 1     0  models.common.Concat [1]
13      -1 2   1182720 models.common.C3  [768, 384, 2, False]
14      -1 1   74112  models.common.Conv [384, 192, 1, 1]
15      -1 1     0  torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
16      [-1, 4] 1     0  models.common.Concat [1]
17      -1 2   296448  models.common.C3  [384, 192, 2, False]
18      -1 1   332160  models.common.Conv [192, 192, 3, 2]
19      [-1, 14] 1     0  models.common.Concat [1]
20      -1 2   1035264 models.common.C3  [384, 384, 2, False]
21      -1 1   1327872 models.common.Conv [384, 384, 3, 2]
22      [-1, 10] 1     0  models.common.Concat [1]
23      -1 2   4134912 models.common.C3  [768, 768, 2, False]
24      [17, 20, 23] 1   24246  models.yolo.Detect [1, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119],
                                                [116, 90, 156, 198, 373, 326]], [192, 384, 768]]
```

YOLOv5M summary: 369 layers, 20871318 parameters, 20871318 gradients, 48.2 GFLOPS

Fonte: Colab

### 4.3 Resultados

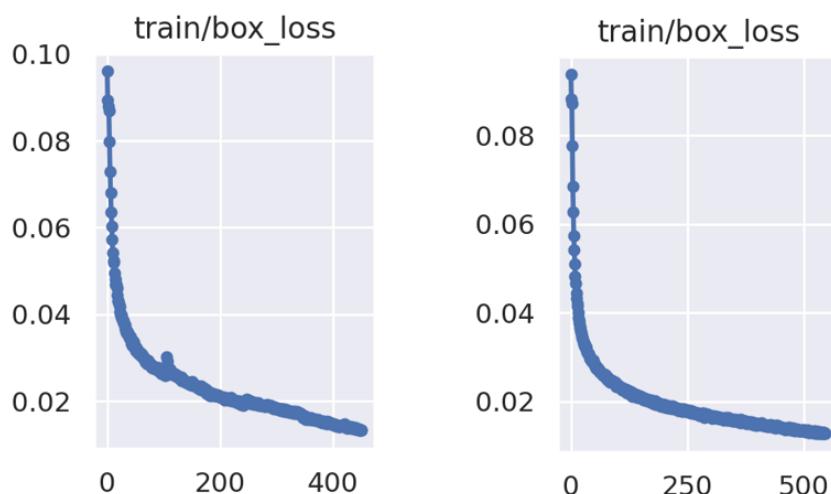
Após extenuantes treinamentos com o *dataset* próprio e uma versão ampliada do mesmo, chegou-se aos resultados que são descritos nos gráficos gerados pelo YOLOv5 e estão dispostos nesta seção.

Na busca por resultados mais rápidos, foi feito a inscrição no *Colab Pro* com o intuito de otimizar o treino e os testes, isso facilitou a utilização das placas de vídeo tornando-as disponíveis mais frequentemente como disponibilizando placas com melhores performances. Além de ampliar o tempo de utilização sem a interferência humana como a quantidade de memória *ram* disponível para o treino.

São expostos dois treinamentos no *dataset* reduzido, sendo um no modelo *small* e um no *medium*. Que são modelos de tamanhos diferentes, apresentando diferentes números de camadas e parâmetros. E um treinamento com o *dataset* ampliado com o modelo *small*. Destacam-se os gráficos de *train/box-loss*, *val/box-loss*, *metrics/precision*, *mAP-0.5* e *mAP-0.5:0.95*. Além de gráficos como *train/obj-loss*, *val/obj-loss* e *metrics/recall*.

Na figura 37 observa-se o gráfico de *train/box-loss*, que determina o quanto as *bound box* estão ajustadas aos objetos verdadeiros no treinamento. Tendo ambos os modelos atingido um patamar equivalente ao final do treino, não possuindo grande diferença entre modelos e *dataset*. O modelo *small* atingindo o valor mínimo de 1,33% e o *medium* 1,27%.

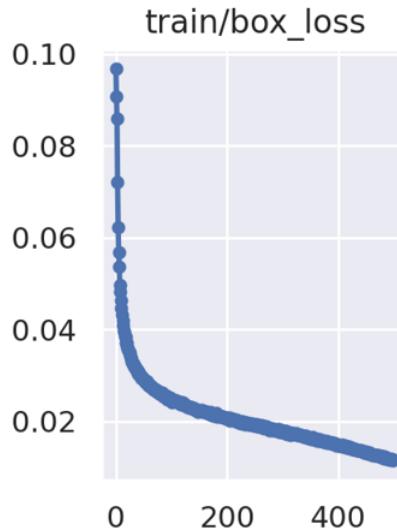
Figura 37 – Comparativo entre modelo *small* na esquerda e *medium* na direita.



Fonte: Colab

Na figura 38 o resultado referente a *train/box-loss* no *dataset* ampliado que atingiu o valor mínimo de 1,15%.

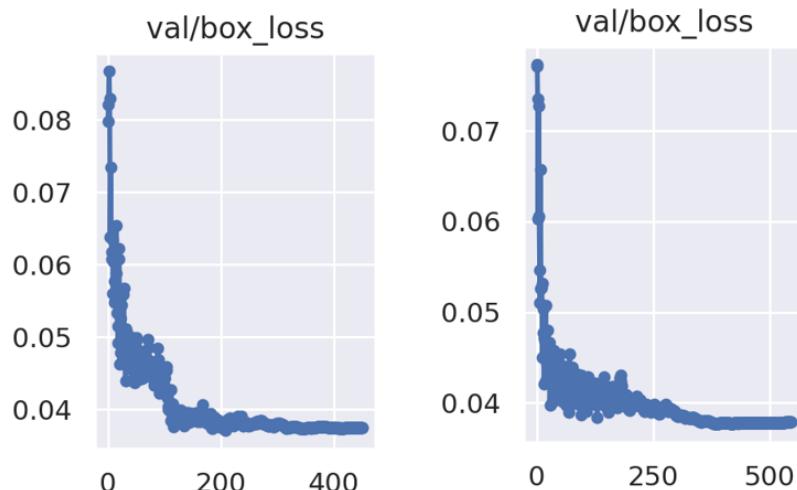
Figura 38 – Gráfico sobre o treinamento no *dataset* ampliado.



Fonte: Colab

Na figura 39 observa-se os gráficos gerados para o parâmetro *val/box-loss* que determina o quanto as *bound box* estão ajustadas aos objetos verdadeiros na validação. Tendo ambos os modelos atingido um patamar equivalente ao final do treino, não possuindo grande diferença entre modelos e *dataset*. O modelo *small* atingindo o valor mínimo de 3,73% e o *medium* 3,77%.

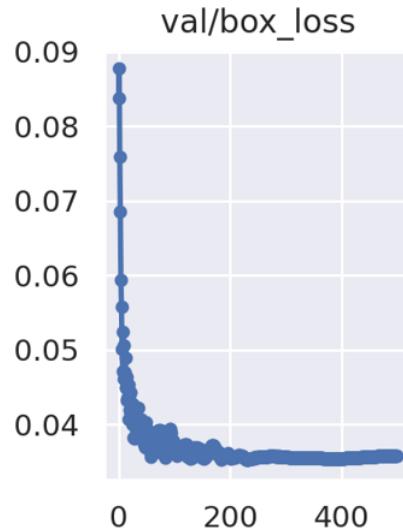
Figura 39 – Comparativo entre modelo *small* na esquerda e *medium* na direita.



Fonte: Colab

Na figura 40 o resultado referente a *val/box-loss* no *dataset* ampliado que atingiu o valor mínimo de 3,53%.

Figura 40 – Gráfico sobre o treinamento no *dataset* ampliado.

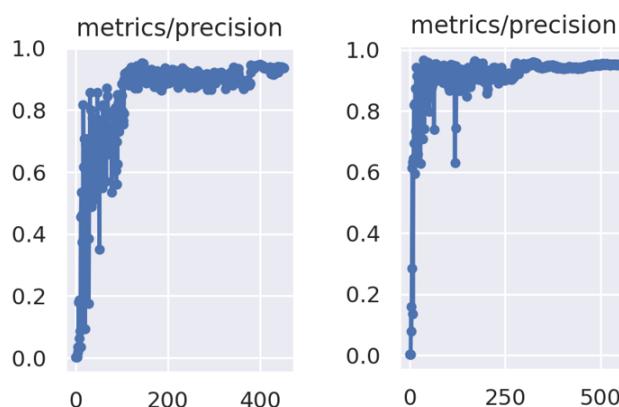


Fonte: Colab

Na figura 41 observa-se o gráfico *metric/precision* que é determinado pela razão entre os *true positive* e a soma entre *true positive* e *false positive*.

Este valor avalia a capacidade do classificador não rotular como positiva uma amostra negativa. Todos os resultados a cerca de *precision* foram satisfatórios, somente o *dataset* ampliado possuindo uma convergência mais suave se comparado aos demais *dataset* e modelos. O modelo *small* atingindo o valor máximo de 94,76% e o *medium* 95,46%.

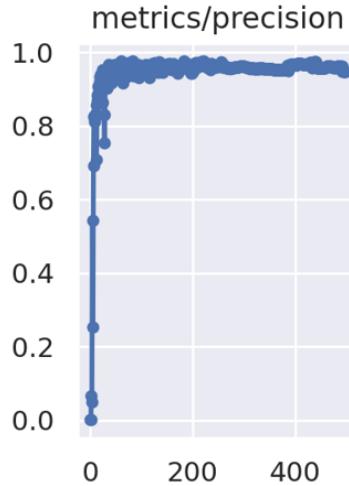
Figura 41 – Comparativo entre modelo *small* na esquerda e *medium* na direita.



Fonte: Colab

Na figura 42 o resultado referente a *metrics/precision* no *dataset* ampliado que atingiu o valor máximo de 97,58%.

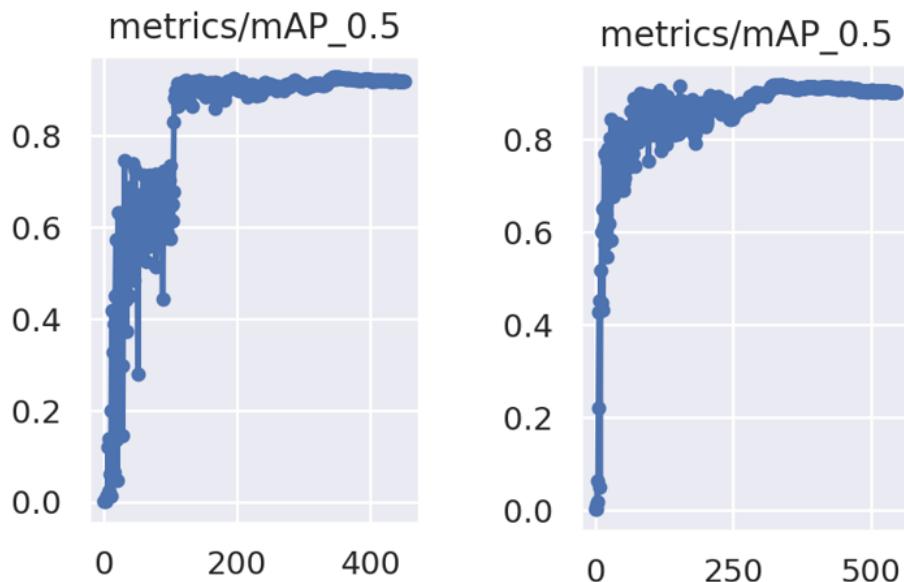
Figura 42 – Gráfico sobre o treinamento no *dataset* ampliado.



Fonte: Colab

Na figura 43 observa-se os resultados de *mean average precision*, destacando que para estar neste gráfico a *IoU*, *Intersection of Union* deve superar os 50%. Neste modelo observa-se que como os demais fatores, o *dataset* ampliado leva vantagem em uma rápida convergência. O modelo *small* atingindo o valor máximo de 92,81% e o *medium* 91,58%.

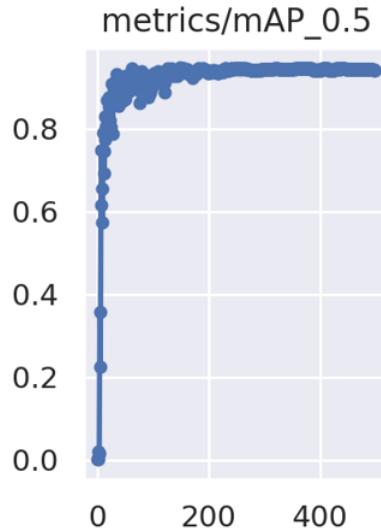
Figura 43 – Comparativo entre modelo *small* na esquerda e *medium* na direita.



Fonte: Colab

Na figura 44 o resultado referente a *mean average precision* no *dataset* ampliado que atingiu o valor máximo de 94,57%.

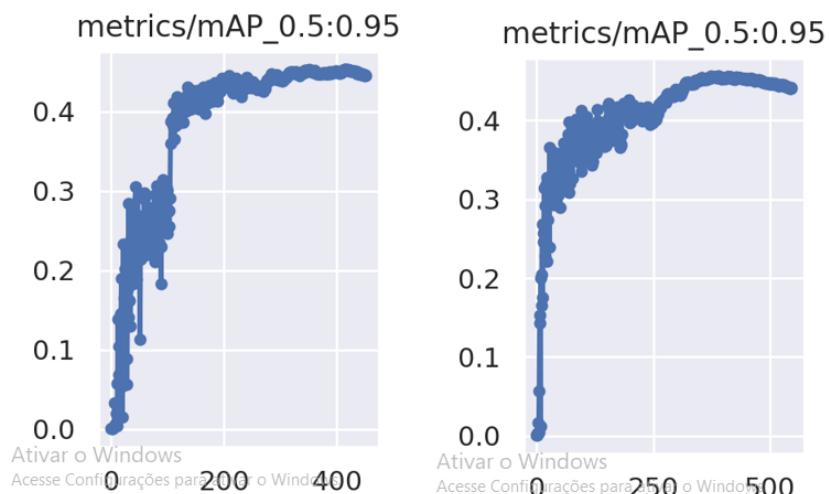
Figura 44 – Gráfico sobre o treinamento no *dataset* ampliado.



Fonte: Colab

Na figura 45 observa-se os resultados de *mean average precision*, destacando que existe uma margem de mudança para a *IoU*, *Intersection of Union* que pode variar de 0.5 até 0.95 com saltos de 0.05. Neste modelo observa-se que como os demais fatores, o *dataset* ampliado leva vantagem em uma rápida convergência. O modelo *small* atingindo o valor máximo de 45,31% e o *medium* 45,63%.

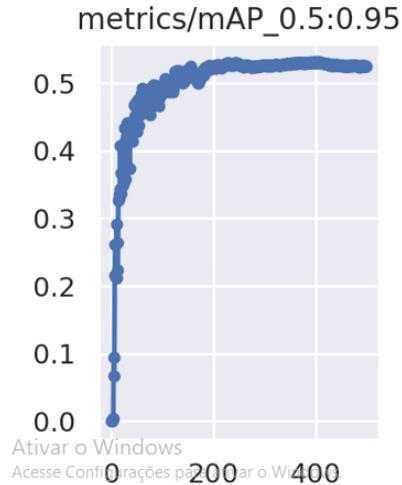
Figura 45 – Comparativo entre modelo *small* na esquerda e *medium* na direita.



Fonte: Colab

Na figura 46 o resultado referente a *mean average precision* 0.5:0.95 no *dataset* ampliado que atingiu o valor máximo de 53,08%.

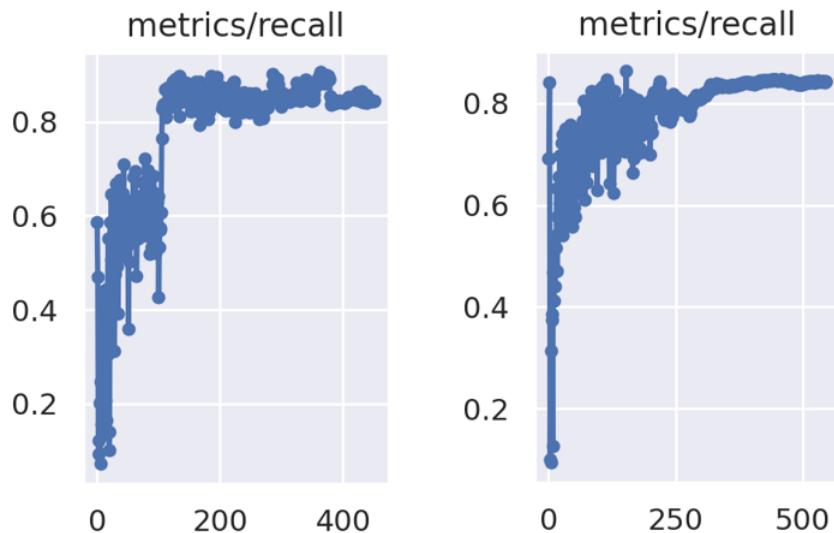
Figura 46 – Gráfico sobre o treinamento no *dataset* ampliado.



Fonte: Colab

Na figura 47 observa-se os gráficos de *recall*, que são oriundo da razão entre os positivos verdadeiros e o número total de positivos, somando os verdadeiros com os falsos. Este gráfico mede a capacidade do modelo de detectar amostras positivas. O modelo *small* atinge o valor máximo de 90,58% e o *medium* 84,77%. O modelo *small* obteve um valor máximo maior que o *medium*, porém, convergiu para 84% igualmente.

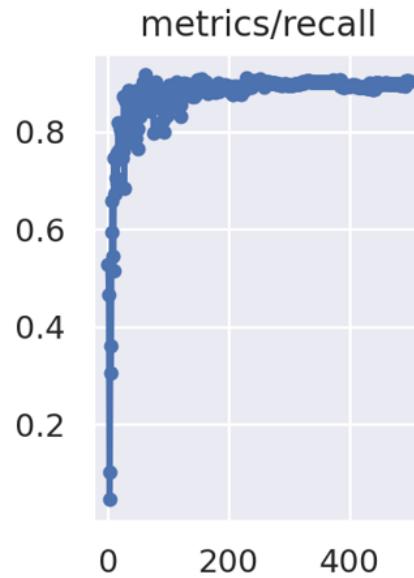
Figura 47 – Comparativo entre modelo *small* na esquerda e *medium* na direita.



Fonte: Colab

Na figura 48 o resultado referente a *recall* no *dataset* ampliado que atingiu o valor máximo de 91,75%.

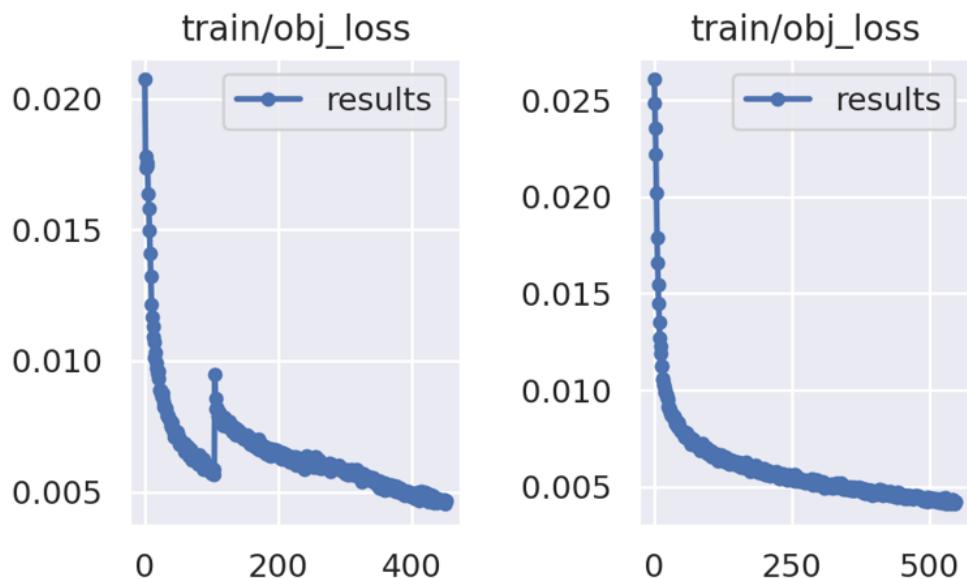
Figura 48 – Gráfico sobre o treinamento no *dataset* ampliado.



Fonte: Colab

Na figura 49 observa-se gráfico *train/obj-loss* que ... O modelo *small* atingiu o valor mínimo de 0,45% e o *medium* 0,41%.

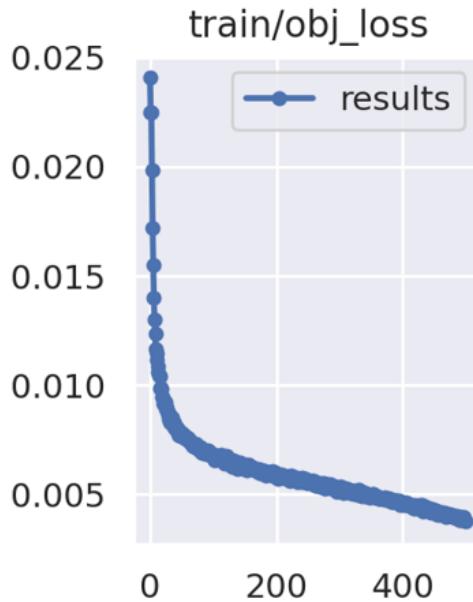
Figura 49 – Comparativo entre modelo *small* na esquerda e *medium* na direita.



Fonte: Colab

Na figura 50 o resultado referente a *train/obj-loss* no *dataset* ampliado que atingiu o valor mínimo de 0,37%.

Figura 50 – Gráfico sobre o treinamento no *dataset* ampliado.



Fonte: Colab

Como pode-se observar no comparativo entre os gráficos, não existe diferença expressiva entre os modelos *small* e *medium* para o *dataset* normal. O *dataset* ampliado chegou a melhores resultados e também atingiu a convergência mais rapidamente. Observa-se na tabela 5 que existe vantagem do *dataset* ampliado em relação ao *dataset* normal e pouca diferença entre os modelos *small* e *medium* com o *dataset* normal. Destaca-se os resultados ótimos do ampliado em *precision*, *recall* e mAP 0.5.

Métrica	DtN-mS	DtN-mM	DtA-mS
Precision	94,76%	95,46%	97,58%
Recall	90,58%	84,77%	91,75%
mAP 0.5	92,81%	91,58%	94,57%
mAP 0.5:0.95	45,31%	46,63%	53,08%
val box-loss	3,73%	3,77%	3,53%
train box-loss	1,33%	1,27%	1,15%
train obj-loss	0,45%	0,41%	0,37%

Tabela 5 – Comparativo de técnicas e *framework* usados.

## 5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Os objetivos foram todos atingidos, desde a criação do *dataset*, adquirindo imagens e rotulando, treino da rede em diferentes modelos, exposição dos gráficos de performance e considerações finais. Mesmo os trabalhos correlatos não possuírem todas as métricas dadas pelo YOLOv5, a tabela 6 demonstra a vantagem nos resultados de performance na comparação deste trabalho e os correlatos. Observa-se que o trabalho possui um bom resultado e pode ser considerado um avanço nos estudos da área da visão computacional. O fator que lhe destaca dos demais é o desenvolvimento do *dataset* próprio, usando imagens de qualidade boa e em diversas cenas com variações de arma, roupa do portador da arma e fundo.

<b>Trabalho</b>	<b>Precision</b>	<b>Recall</b>	<b>mAP 0.5</b>
Romero e Salamea (2019)	86%	86%	
Bhatti et al. (2021)	85,52%	85,98%	91,73%
SINGH et al. (2021)			77,75%
Autor próprio	97,58%	91,75%	94,57\$

Tabela 6 – Comparativo de trabalhos e seus resultados.

Portanto o modelo é capaz de realizar seu objetivo inicial, detectar armas em imagens de câmeras de segurança, entretanto, necessita de câmeras ainda melhores para a aquisição das imagens tanto para o treino quanto para a detecção, de preferência que gravem em 4k e 60 fps ou melhor. Tornando-se evidente a necessidade de criar um estúdio em um clube de tiro, onde existem diversas pessoas portando e manuseando as armas. Através dessas atividades inerentes ao clube, pode-se extrair essas imagens para o desenvolvimento de um *dataset* robusto. Vale destacar a necessidade de um acerto com os atores nas cenas captadas, pois existe o interesse em tornar público este *dataset* em algum repositório, devido aos problemas com direito de imagem. Outro ponto a destacar é o interesse na criação de um grupo de pesquisa com o intuito de avançar no desenvolvimento de tecnologias no setor da segurança juntamente com a visão computacional. Para trabalhos futuros vale ressaltar a utilização do YOLOv5n6 e YOLOv5s6 que são variantes do YOLOv5 que aceitam como entrada imagens maiores, aumentando assim a quantidade de informação para ser analisada.

## REFERÊNCIAS

- ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. In: IEEE. **2017 international conference on engineering and technology (ICET)**. [S.I.], 2017. p. 1–6.
- BHATTI, M. T. et al. Weapon detection in real-time cctv videos using deep learning. **IEEE Access**, IEEE, v. 9, p. 34366–34382, 2021.
- BHUYAN, M. K. **Computer Vision and Image Processing: Fundamentals and Applications**. [S.I.]: CRC Press, 2019.
- BITTENCOURT, G. **Inteligência artificial: ferramentas e teorias**. [S.l.: s.n.], 1998.
- BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. **arXiv preprint arXiv:2004.10934**, 2020.
- CELEBI, M.; CEYLAN, M. The new activation function for complex valued neural networks: Complex swish function. In: **Proceedings of the 4th International Symposium on Innovative Approaches in Engineering and Natural Sciences, Samsun, Turkey**. [S.I.: s.n.], 2019. p. 22–24.
- FUKUSHIMA, K. Biological cybernetics neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. **Biol Cybern**, v. 36, p. 193–202, 1980.
- GARDNER, H.; VERONESE, M. A. V. **Inteligências múltiplas: a teoria na prática**. [S.I.]: Artes Médicas, 1995.
- HAO, X.; ZHANG, G.; MA, S. Deep learning. **International Journal of Semantic Computing**, World Scientific, v. 10, n. 03, p. 417–439, 2016.
- HASSAN, M. ul. Vgg16-convolutional network for classification and detection. en línea].[consulta: 10 abril 2019]. Disponible en: <https://neurohive.io/en/popular-networks/vgg16>, 2018.
- HAYKIN, S. **Redes neurais: princípios e prática**. [S.I.]: Bookman Editora, 2007.
- HUSSEIN, B. et al. **Application of Computer Vision and Machine Learning for Digitized Herbarium Specimens: A Systematic Literature Review**. 2021.
- LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.
- LI, W. et al. Frd-cnn: Object detection based on small-scale convolutional neural networks and feature reuse. **Scientific reports**, Nature Publishing Group, v. 9, n. 1, p. 1–12, 2019.
- LIU, S. et al. Path aggregation network for instance segmentation. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.I.: s.n.], 2018. p. 8759–8768.

- LIU, W. et al. Ssd: Single shot multibox detector. In: SPRINGER. **European conference on computer vision**. [S.l.], 2016. p. 21–37.
- MEDEIROS, L. F. de. **Inteligência artificial aplicada: uma abordagem introdutória**. [S.l.: s.n.], 2018. (1).
- MINSKY, M. L. **La sociedad de la mente : la inteligencia humana a la luz de la inteligencia artificial**. [S.l.: s.n.], 1986. ISBN 9509480029.
- MISRA, D. Mish: A self regularized non-monotonic neural activation function. **arXiv preprint arXiv:1908.08681**, CoRR, v. 4, p. 2, 2019.
- NORVIG, S. R. P. **Inteligência Artificial**. 3rd. ed. [S.l.]: Elsevier, 2013. ISBN 978-8535237016.
- PALAZZO, L. A. **Introdução à programação Prolog**. [S.l.]: EDUCAT, Editora da Universidade Católica de Pelotas, 1997.
- REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. **arXiv preprint arXiv:1804.02767**, 2018.
- REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. **Advances in neural information processing systems**, v. 28, p. 91–99, 2015.
- REZENDE, S. O. **Sistemas inteligentes: fundamentos e aplicações**. [S.l.]: Editora Manole Ltda, 2003.
- ROMERO, D.; SALAMEA, C. Convolutional models for the detection of firearms in surveillance videos. **Applied Sciences**, Multidisciplinary Digital Publishing Institute, v. 9, n. 15, p. 2965, 2019.
- SINGH, A. et al. Iot based weapons detection system for surveillance and security using yolov4. In: IEEE. **2021 6th International Conference on Communication and Electronics Systems (ICCES)**. [S.l.], 2021. p. 488–493.
- STERNBERG, R. J. **Psicología Cognitiva**. 5th. ed. [S.l.]: Cengage, 2010.
- SZEGEDY, C. et al. Inception-v4, inception-resnet and the impact of residual connections on learning. In: **Thirty-first AAAI conference on artificial intelligence**. [S.l.: s.n.], 2017.
- SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 2818–2826.
- XU, B. et al. Empirical evaluation of rectified activations in convolutional network. **arXiv preprint arXiv:1505.00853**, 2015.
- XU, R. et al. A forest fire detection system based on ensemble learning. **Forests**, v. 12, p. 217, 02 2021.
- ZOU, F. et al. A sufficient condition for convergences of adam and rmsprop. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2019. p. 11127–11135.