UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Cícero Pereira de Lima Junior

Estudo de viabilidade de sistemas de detecção de armamentos em tempo real em linhas de ônibus urbanos

Recife

2021

Cícero Pereira de Lima Junior

Estudo de viabilidade de sistemas de detecção de armamentos em tempo real em linhas de ônibus urbanos

Monografia apresentada ao Curso de Bacharelado em Ciências da Computação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Ciências da Computação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Computação

Curso de Bacharelado em Ciências da Computação

Orientador: Douglas Véras e Silva

Recife

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

L732e Lima Júnior, Cícero Pereira de
Estudo de viabilidade de sistemas de detecção de armamentos em tempo real em linhas de ônibus urbanos / Cícero Pereira de Lima Júnior. - 2021.
68 f. : il.

Orientador: Douglas Veras e Silva.
Inclui referências e apêndice(s).

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Ciência da Computação, Recife, 2021.

1. Visão Computacional. 2. Redes Neurais. 3. Aprendizagem Profunda. 4. Sistemas Distribuídos. 5. Análise de Desempenho. I. Silva, Douglas Veras e, orient. II. Título

CDD 004

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

http://www.bcc.ufrpe.br

Ficha de Aprovação do Trabalho de Conclusão de Curso

Trabalho defendido por Cícero Pereira de Lima Junior às 15 horas do dia 09 de dezembro de 2021, no link https://meet.google.com/zjw-tgsd-pyp, como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado "Estudo de viabilidade de sistemas de detecção de armamentos em tempo real em linhas de ônibus urbanos", orientado por Douglas Véras e Silva e aprovado pela seguinte banca examinadora:

Douglas Véras e Silva
DC/UFRPE

Gustavo Rau de Almeida Callou
DC/UFRPE

Dedico este trabalho aos meus pais, pois sem eles nada disso seria possível.

Agradecimentos

A Deus, pela minha vida e por me permitir superar todos os obstáculos.

A todos os professores da Universidade Federal Rural de Pernambuco que contribuíram para a minha formação até aqui.

Ao professor Ricardo Souza, pela valiosa oportunidade de trabalhar em um projeto de extensão.

Ao orientador Douglas Véras, pela paciência, sabedoria e por motivar o desenvolvimento deste trabalho.

À secretária Sandra Xavier por tornar todo o processo, desde o ingresso, mais fácil.

À minha companheira, Stephanie Mota, por ter acreditado em mim do início ao fim.

A todos os amigos que fiz durante todos estes anos de curso, em especial André Lins, Carlos Olímpio, César Muniz e Ramicés Moisés, pelos diversos trabalhos e horas de estudo.

A todos meus colegas do C.E.S.A.R, que me apoiaram e incentivaram a finalizar minha graduação.

Aos funcionários do Restaurante Universitário e aos responsáveis pelos estabelecimentos Espaço Vivência e Conterrâneo.

Ao meu amigo, Marlon Sukar, por me ajudar com a revisão deste trabalho e pelos 16 anos de amizade.

“Um mago nunca chega atrasado, nem adiantado. Ele chega exatamente quando quer chegar.”

(Gandalf, Em O Senhor dos Anéis, Peter Jackson)

Resumo

Sistemas de monitoramento são parte fundamental na prevenção de assaltos a mão armada dentro dos ônibus. Entretanto, a utilização desses sistemas demanda quantidade irreal de pessoas para que seja operado em tempo real. O uso de técnicas de visão computacional e aprendizagem profunda surgem como forma de automatizar partes ou até todo o processo de monitoramento, desde a detecção dos armamentos ao disparo de alarmes. Para que este processo seja realizado de maneira eficiente, permitindo ações efetivas por parte das autoridades, é preciso que o sistema seja capaz de atender em tempo real uma demanda crescente de câmeras de segurança. Desta forma, este trabalho teve como objetivo analisar a viabilidade de um sistema de detecção de armamentos em uma rede de ônibus. Através de simulação, avaliou-se o desempenho do algoritmo YOLO, em sua quarta versão, em um modelo cliente-servidor sob uma demanda crescente de câmeras de segurança. O servidor dispunha de uma GPU Tesla V80 com 12GB de memória, processador Intel Xeon dual core, 61GB de memória RAM e 200GB de espaço em disco. Por fim, a partir dos resultados obtidos, observou-se que a aplicação apresenta aumento no tempo de detecção após introduzir 16 usuários virtuais (câmeras) e seu tempo médio não pode ser considerado como tempo real, dentro do contexto de segurança de ônibus.

Palavras-chave: Visão Computacional, Redes Neurais, Aprendizagem Profunda, Sistemas Distribuídos, Análise de Desempenho.

Abstract

Surveillance systems are fundamental on preventing armed robberys on public busses. However, to be operated in real-time theses systems demand an unrealistic amount of people. The usage of computer vision and deep learning technics raises as a way to automate parts or even the whole surveillance process, from the weapons detection to the alarm triggering. For this process to be accomplished efficiently, allowing authorities to take more effective actions, the system needs to be able to handle a growing security cameras demand. Thus, this work analyses a bus line weapon detection system viabillity. Through simulation, this work evaluated the perfomance of YOLO algorithm, in its fourth version, on a client-server model under a growing security camera demand. The server is composed of a Tesla V80 GPU with a 12GB memory, Intel Xeon dual core processor, 61GB RAM memory and 200GB disk space. Finally, from the gathered results, its observable that the application presents a detection time increase after having introduced 16 virtual users (cameras), also the average response time cannot be considered as real-time, on bus security context.

Keywords: Computer Vision, Neural Networks, Deep Learning, Distributed Systems, Performance Evaluation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AP	Average Precision
API	Application Program Interface
CFTV	Circuito Fechado de Televisão
CNN	Convolutional Neural Networks
DDNN	Distributed Deep Neural Networks
FN	False Negative
FP	False Positive
IoT	Internet of Things
IOU	Intersection over Union
mAP	Mean Average Precision
NVR	Network Video Recorder
ROS	Robot Operating System
RPA	Region Proposal Algorithm
RPN	Region Proposal Networks
SPP	Spatial Pyramide Pooling
SSD	Single-shot Multibox Detector
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
YOLO	You Only Look Once

Sumário

1 Introdução

No ano de 2020, em decorrência do agravamento da crise econômica[1], crimes como assaltos a ônibus vêm se tornando cada vez mais frequentes em alguns países latino-americanos. Em Bogotá, capital colombiana, cerca de 400 pessoas foram detidas por realizar delitos em ônibus do sistema de transporte público TransMilenio (EL TIEMPO, 2021). Crimes desta natureza também têm sido registrados de maneira crescente em cidades de países como Peru[2] e Paraguai[3]. O problema também atinge de maneira preocupante o México, que no ano de 2019, até o mês de agosto, havia registrado mais de 12845 casos de roubos a ônibus, dentre estes, 9358 foram realizados com uso de violência (QUEZADA, 2019).

No Brasil, a violência e segurança pública estão entre as quatro principais preocupações dos cidadãos, como revela pesquisa realizada pela Conferência Nacional das Indústrias (CNI, 2018), tendo sido citado por 38% dos participantes. Os crimes no país, dentro do transporte público, ocorrem de maneira mais recorrente em grandes capitais. De acordo com dados da Secretaria de Segurança Pública da Bahia[4], foram registradas, até junho de 2021, 827 ocorrências de roubos a coletivos, sendo 691 apenas na capital, Salvador. Na capital do Rio de Janeiro, a incidência de crimes aos coletivos apresentou um aumento de 105.1%, quando comparados os meses de abril de 2020 e abril de 2021 (G1, 2021).

Na região metropolitana do Recife, apesar de demonstrarem queda a partir do ano de 2018, os números de assaltos a ônibus ainda são preocupantes[5]. Segundo dados da Secretaria de Defesa do estado de Pernambuco[6], 629 ocorrências foram registradas no ano de 2020, sendo 328 na região metropolitana, como mostra a Tabela 1.

1 <https://www.elsoldetoluca.com.mx/local/asaltos-en-el-transporte-publico-han-aumentado-pese-a-contingencia-canapat-5203887.html>
2 <https://www.americatv.com.pe/noticias/actualidad/aumentan-robos-buses-transporte-publico-n445944>
3 <https://www.abc.com.py/nacionales/2021/09/08/policia-sube-a-los-colectivos-para-realizar-controles-tras-ola-de-asaltos/>
4 <http://www.ssp.ba.gov.br/modules/conteudo/conteudo.php?conteudo=124>
5 <https://jc.ne10.uol.com.br/colunas/mobilidade/2021/05/12125597-assaltos-a-onibus-tem-aumento-no-grande-recife-e-em-todo-estado-de-pernambuco.html>
6 <https://www.sds.pe.gov.br/estatisticas/40-estatisticas/177-crimes-violentos-contra-o-patrimonio>

Tabela 1 – Série histórica de roubos de ônibus em Pernambuco

Região	2014	2015	2016	2017	2018	2019	2020
Capital	208	392	536	650	411	334	289
Região Metropolitana	291	413	687	705	454	484	328
Interior	25	36	49	56	50	19	17
Pernambuco	524	841	1272	1411	915	837	629

Fonte: Secretaria de Defesa Social - PE

Os índices de criminalidade apresentados, afetam de maneira substancial a qualidade do transporte público. Segundo (NEWTON, 2014), o transporte público tem um papel fundamental no combate à exclusão social, uma vez que facilita o acesso a atividades fundamentais como saúde, empregos e lazer. Entretanto, como afirma (CARDOSO; SANTOS; SILVA, 2021), o medo atrelado à violência e segurança pessoal são fatores inibidores do transporte público e que devem ser levados em consideração ao tentar compreender a eficácia do transporte público, pois esses fatores diminuem a confiança no serviço prestado, bem como a acessibilidade ao transporte, propriamente dito e a outros serviços essenciais.

Os sistemas de monitoramento por CFTV (circuito fechado de televisão) são peça fundamental na prevenção de crimes desta natureza, e vêm tornando-se mais acessíveis ao público e se sofisticando para uso policial (DAVIES; VELASTIN, 2014; BARBASCHOW, 2021). Entretanto, sistemas de monitoramento passivos, exigem alta supervisão humana, sendo muitas vezes necessário que um único operador monitore de 16 até 78 câmeras simultaneamente (DEE; VELASTIN, 2008; BRUIJN et al., 2015), podendo se tornar caro e pouco efetivo (LAI; MAPLES, 2017).

Com os avanços observados na área de inteligência artificial a partir do ano de 2014, em especial, nas redes neurais e aprendizagem profunda, abriu-se uma gama de novas aplicações para a detecção de objetos (ZOU et al., 2019). O surgimento de algoritmos de detecção de objetos baseados em redes neurais convolucionais, permitiu que tarefas ligadas à detecção de objetos fossem realizadas de maneira mais rápida e robusta, juntamente com o uso de técnicas de aprendizagem de máquina multiescala, *data augmentation*, detecção baseada em contexto e redes adversárias generativas, que podem tornar mais eficientes as detecções de objetos pequenos em imagens amplas (TONG; WU; ZHOU, 2020).

Todavia, a implantação de um sistema de larga escala para detecção de objetos em tempo real tende a ser custosa, pois o volume de dados produzidos demanda altas

taxas de transmissão, sendo comum a utilização de redes de fibra óptica para tratar a alta vazão (REN et al., 2018). No caso particular de uma rede de ônibus, o envio de um alto volume de dados introduz um alto tráfego na rede, ocasionando um alto consumo de largura de banda (CUI et al., 2021)

Tendo em vista as taxas de crimes e as limitações do uso de sistemas de monitoramento passivos a fim de frear a ação de criminosos, junto com os avanços da Inteligência Artificial e Visão Computacional, surge a possibilidade da aplicação de algoritmos destinados à detecção automática de armamentos em imagens de circuitos fechados como alternativa ou complemento ao trabalho dos operadores em centrais de monitoramento, permitindo uma detecção de crimes de maneira mais assertiva e possibilitando ações mais rápidas por parte das autoridades (LIM et al., 2019).

1.1 Justificativa

Os diversos trabalhos relacionados a temática de detecção de armamentos, focam em dois diferentes aspectos: trabalhos como o de (OLMOS; TABIK; HERRERA, 2018) e (WARSI et al., 2019) são focados na implementação de técnicas de detecção de objetos em vídeos de segurança, sem se ater aos aspectos atrelados a sua aplicação em larga escala. Já trabalhos como o de (Cheng; Sun; Tu, 2018) e (HERRERA et al., 2018), utilizam-se de sistemas distribuídos visando melhorias no tempo de execução de detectores.

Embora os trabalhos tenham apresentado resultados promissores, em termos de qualidade de detecção e tempo de detecção, suas avaliações foram realizadas em função do modelo de aprendizagem profunda, sem que sua eficácia tenha sido testada em larga escala. Mesmo trabalhos como o de (ROHIT, 2020), que visam soluções destinadas ao transporte público, ainda apresentam limitações quanto a eficiência de seu sistema de detecção de armas em tempo real.

Desta forma, será realizado um estudo que visa simular a implantação de um sistema de detecção de objetos dentro do transporte público. Portanto, este trabalho analisará o desempenho do algoritmo de detecção de objetos YOLOv4 (*You Only Look Once*) (BOCHKOVSKIY; WANG; LIAO, 2020), devido ao sua velocidade de detecção superior as suas versões anteriores e ao SSD (*Single-shot Multibox Detector*) (LIU et al., 2016), em um modelo cliente-servidor a fim de identificar se este é capaz de atender uma demanda crescente de câmeras e quais são suas limitações.

1.2 Problema

Imagens captadas por câmeras de ônibus apresentam cenas complexas, podendo conter a presença de diferentes objetos em uma única imagem. Considerando este fato, a detecção de armas em uma destas cenas é uma tarefa complexa, dadas as limitações dos modelos de aprendizagem profunda atuais na detecção de objetos pequenos em imagens amplas. Outro fator determinante para a assertividade das detecções, é o possível viés introduzido durante o treinamento de um modelo, uma vez que alguns objetos, como aparelhos celulares, apresentem tamanhos e posições similares a de uma arma em cena, fator que pode ser agravado pela qualidade das imagens.

A infraestrutura necessária para a implantação de um sistema de detecções de armas em tempo real em ônibus introduz uma segunda problemática: a quantidade de imagens captadas pelas câmeras de múltiplos ônibus produz um alto volume de dados, consequentemente, produzindo gargalos e alto consumos de banda na rede.

Considerando estas duas maiores problemáticas, este trabalho busca responder a seguinte pergunta:

“Qual a infraestrutura necessária para a aplicação de técnicas de detecção de armamentos em imagens de circuito fechado em tempo real de maneira escalável e eficiente?”

1.3 Objetivos

Nesta seção serão expostos os objetivos almejados com a realização deste trabalho.

1.3.1 Objetivo geral

Simular um sistema de detecção de armamentos em linhas de ônibus, a fim de avaliar sua capacidade de operação em tempo real e sua escalabilidade.

1.3.2 Objetivos específicos

1. Criar a partir de bases de dados distintas um novo conjunto de dados de imagens e anotações para dois tipos de armamentos: Armas e Facas.
2. Treinar e avaliar, a partir de um conjunto de dados, um modelo de aprendizagem profunda capaz de identificar armamentos em imagens de circuito fechado.

3. Indicar algoritmos e infraestrutura necessária para a implantação de um sistema real.

1.4 Organização do trabalho

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta aspectos teóricos referentes à visão computacional e sistemas distribuídos, bem como métricas de avaliação de desempenho; o Capítulo 3 abordas os trabalhos relacionados que propõem técnicas de detecção de armamentos e arquiteturas distribuídas; o Capítulo 4 aborda a metodologia utilizada para desenvolver e avaliar as técnicas utilizadas; o Capítulo 5 apresenta os resultados obtidos após execução do experimento; por fim, o Capítulo 6 apresenta as conclusões deste trabalho, assim como suas limitações e trabalhos futuros.

2 Fundamentação teórica

Neste capítulo serão expostas noções e princípios fundamentais para a compreensão e fundamentação do tema abordado neste trabalho.

Este capítulo está dividido em três seções: A Seção 2.1 discorre sobre conceitos básicos sobre imagem e visão computacional, introduzindo também o conceito de redes neurais e redes neurais convolucionais para melhor compreensão do algoritmo YOLOv4, abordado na Subseção 2.1.3.3.2. A Seção 2.2 aborda conceitos de sistemas distribuídos, enquanto a Seção 2.3 apresenta métricas utilizadas tanto em modelos de aprendizagem profunda voltados à detecção de objetos, como métricas e abordagens aplicadas à avaliação de desempenhos de serviços.

2.1 Visão Computacional

Segundo (PETERS, 2017), o objetivo da Visão computacional, é interpretar e reconstruir cenas naturais computacionalmente, baseando-se no conteúdo de imagens capturadas através de câmeras digitais. Cenas naturais, por sua vez, são a parte do campo visual que pode ser capturada pelos olhos humanos ou sensores ópticos. Em outras palavras, a Visão Computacional se propõe a descrever o mundo visível e reconstruir suas propriedades, tais como iluminação, forma e distribuição de cores (SZELISKI, 2010). Com base nestas definições, podemos entender a Visão Computacional como um processo que automatiza as funções cognitivas desempenhadas pelos olhos.

A Visão Computacional é aplicada em uma gama de aplicações do mundo real (SZELISKI, 2010; DAVIES, 2012), algumas destas são:

- Reconhecimento óptico de caracteres: aplicada à leitura de códigos postais manuscritos (NIVETHA et al., 2021; MITHE; INDALKAR; DIVEKAR, 2013) e reconhecimento automático de placas veiculares (LEE et al., 2017).
- Inspeção de Máquinas: garantia de qualidade de peças através da inspeção rápida utilizando técnicas de visão estéreo com iluminação especializada para medir tolerância a falhas nas asas de aeronaves ou busca por defeitos em peças metálicas com uso de visão de raio-X (MALEKZADEH et al., 2017).
- Modelagem 3D: construção completa e automática de modelos tridimensionais a partir de imagens aéreas, comumente utilizadas em aplicações como mapas (YAMAZAKI et al., 2015).

- Exames de imagem: processamento em imagens obtidas através de exames como ultrassonografia e tomografia visando extrair informações voltadas ao diagnóstico de doenças (DOMINGUES et al., 2019).
- Vigilância e monitoramento: usado para identificar invasões em áreas restritas (BHANDARI; PARK, 2020), monitoramento de tráfego em avenidas (LIU et al., 2021) e monitoramento de piscinas para identificar possíveis vítimas de afogamento (SALEHI; KEYVANARA; MONADJEMMI, 2016).

2.1.1 Imagem digital

Imagem digital é uma representação discreta de um objeto do campo visual que possui informações de espaço (arranjo) e intensidade (cores ou escala de cinza) (PETERS, 2017).

Partindo de uma definição visual, uma imagem digital em escala de cinza é representada por uma função bidimensional $\boldsymbol{I}(x, y)$, onde x e y representam as coordenadas espaciais e o valor de $\boldsymbol{I}$ em (x, y) é proporcional a intensidade de luz captada por um sensor óptico e gravada em seu correspondente elemento de imagem(Pixel) naquele determinado ponto. Em uma imagem digital em cores, por sua vez, um pixel em (x, y) é uma matriz 1x3 onde cada elemento indica a intensidade de Vermelho, Verde ou Azul do pixel em um canal de cor (PETERS, 2017).

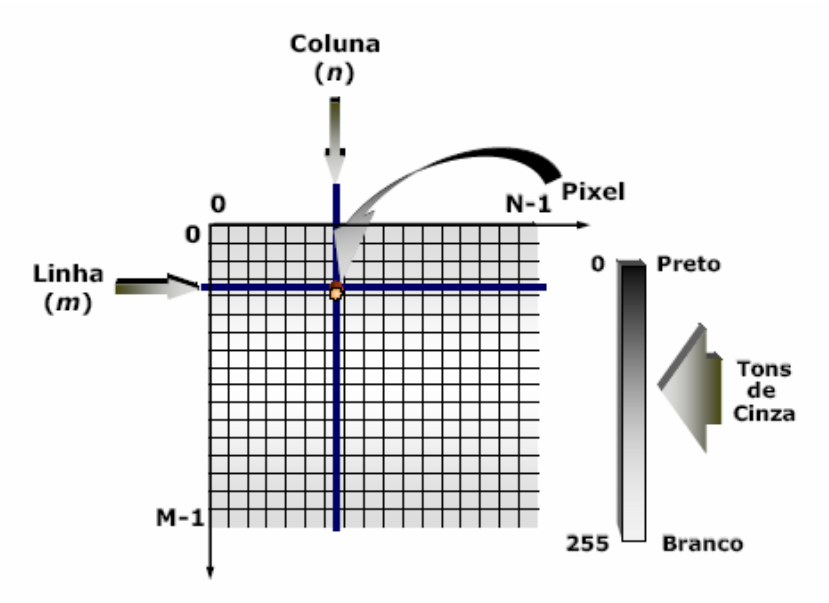
Uma imagem digital, portanto, pode ser compreendida como uma malha bidimensional onde cada um dos pontos são denominados de *pixels* ou elementos de imagem. A Figura 1 ilustra a representação computacional de uma imagem de duas dimensões em escala de cinza.

2.1.2 Detecção de objetos em imagens

O objetivo da detecção de objetos é identificar instancias de objetos pertencentes a uma classe conhecida, como pessoas, carros ou rostos em uma imagem. Tipicamente, apenas um pequeno número de instâncias de um objeto está presente na imagem, mas há uma gama de possibilidades de posicionamento e tamanho que podem ocorrer e necessitam, de alguma maneira, ser exploradas (AMIT; FELZENSZWALB; GIRSHICK, 2020).

Cada detecção é reportada através de algum formato que expresse informações sobre a posição do objeto. Estes formatos podem se apresentar simplesmente como a localização do objeto, localização e escala ou pela extensão do objeto descrita em termos de uma caixa delimitadora (AMIT; FELZENSZWALB; GIRSHICK, 2020), como mostrada na Figura 2.

Figura 1 – Representação de uma imagem digital em escala de cinza



Fonte: (SOUZA; CORREIA, 2021)

Figura 2 – Detecção de objeto representada por caixa delimitadora.

Fonte: Adaptado de (KUMAR, 2020)

Ao longo dos últimos 20 anos e após o surgimento da aprendizagem de máquina e aprendizagem profunda, a detecção de objetos vem sendo utilizada em uma serie de aplicações (ZOU et al., 2019), das quais podemos destacar:

- Detecção de pedestres: utilizada para monitoramento de tráfego, investigações criminais e carros autônomos (DALAL; TRIGGS, 2005; DOLLáR et al., 2009).

- Detecção de faces: umas das aplicações mais antigas da detecção de objetos, sendo utilizada em câmeras digitais para detecção de sorrisos, troca de rostos e até aplicação de filtros de maquiagem em aplicativos móveis (VIOLA; JONES, 2004).
- Leitura de textos: visa identificar caracteres contidos em imagens digitais permitindo a sua conversão em texto (WANG; BABENKO; BELONGIE, 2011).
- Identificação de sinais de trânsito: detecção automática de sinais de trânsito em aplicações para carros autônomos (BAHLMANN et al., 2005).
- Sensoriamento remoto: utilizado em investigações militares, gerenciamento de tráfego urbano e resgate de vítimas de desastres (CAI et al., 2018).

2.1.3 Algoritmos de detecção

Segundo (ZOU et al., 2019), os avanços na detecção de objetos em imagens podem ser genericamente divididos em dois períodos históricos: o período das técnicas tradicionais de detecção (pré 2014) e o período das técnicas baseadas em aprendizagem profunda (pós 2014). Os algoritmos de detecção citados neste trabalho surgiram a partir do segundo período histórico. Desta forma, nas subseções 2.1.3.1, 2.1.3.2 e 2.1.3.3, serão introduzidos conceitos básicos de aprendizagem profunda para a melhor compreensão destes algoritmos.

2.1.3.1 Redes neurais

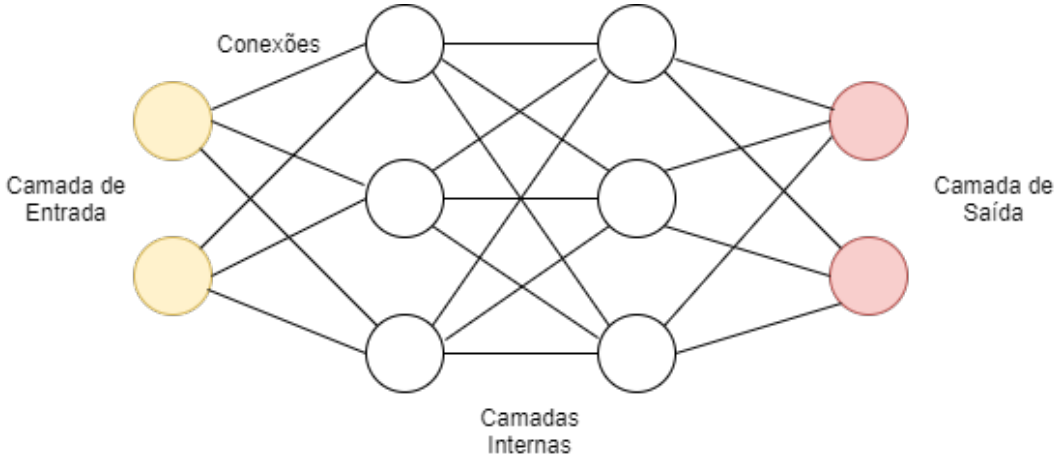
As redes neurais artificiais, ou simplesmente redes neurais, são estruturas criadas baseadas no funcionamento do cérebro humano e sua interconexão de neurônios. Uma rede neural, consiste em uma camada de neurônios de entrada (também chamados de nós ou unidades), uma a três camadas de neurônios internos e uma camada de neurônios de saída (WANG, 2003).

Um neurônio, em uma rede neural artificial, consiste em um uma função matemática que coleta e classifica informações de acordo com uma arquitetura específica (CHEN, 2020).

2.1.3.2 Aprendizagem profunda

Segundo (BENGIO; GOODFELLOW; COURVILLE, 2017), aprendizagem profunda é um tipo de aprendizagem de máquina que obtém grande poder e flexibilidade aprendendo a representar o mundo como uma hierarquia aninhada de conceitos e representações, com cada conceito sendo definido com relação a conceitos mais simples

Figura 3 – Arquitetura de uma rede neural artificial *feed-forward*.



Fonte: Adaptado de (Data Science Academy, 2021)

e representações mais abstratas computadas em termos de representações menos abstratas.

Aprendizagem profunda consiste em redes neurais multicamadas, onde a informação é transmitida através de cada camada, de forma que a saída de uma camada forneça a entrada para a camada subsequente. A primeira camada é chamada de camada de entrada, enquanto a última camada é chamada de camada de saída. Todas as demais camadas, entre a camada de entrada e a de saída, são chamadas de camadas ocultas ou internas (Data Science Academy, 2021).

2.1.3.3 Redes neurais convolucionais

As redes neurais convolucionais ou CNN (*Convolutional Neural Networks*) são algoritmos de aprendizagem profunda capazes de aprender características a partir do treinamento. Uma rede neural convolucional é capaz de, a partir de uma imagem de entrada, atribuir pesos a aspectos (objetos presentes na imagem) e realizar a distinção uns dos outros (Data Science Academy, 2021).

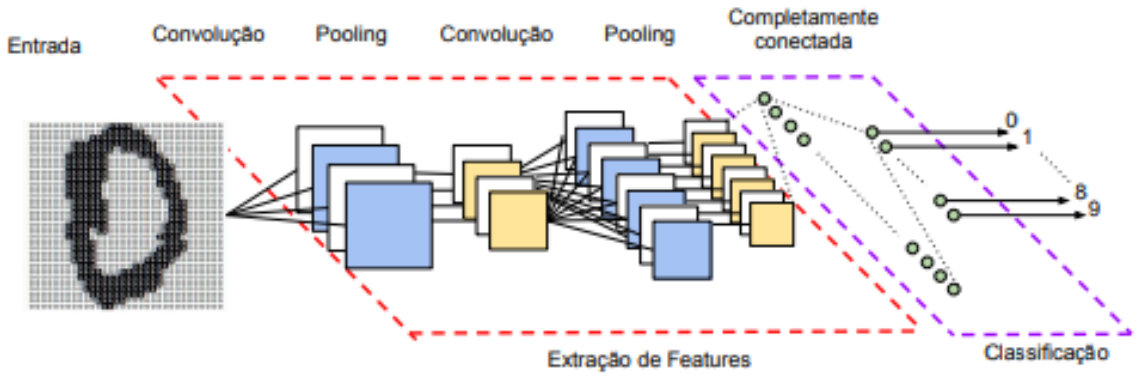
Como mostrado na Figura 4, uma rede neural convolucional apresenta múltiplas partes que desempenham funções distintas. As primeiras camadas apresentadas na rede são chamadas de camadas de convolução. Os neurônios, ou nós, presentes nessas camadas são responsáveis pela realização de funções tradicionais em visão computacional, como aplicação de filtros, sendo também responsáveis pela extração de características, gerando um mapa de características, mais conhecido como *feature map*.

Após as camadas convolucionais, existem as camadas de *pooling*. Estas camadas são responsáveis pela redução espacial de uma imagem, combinando os *pixels* de

vizinhança de uma determinada região da imagem em um valor representativo único (KIM, 2017). Está camada é de grande importância, tanto para agilizar o processo de treinamento da rede neural, como para prover invariância espacial (VARGAS; PAES; VASCONCELOS, 2016).

Em redes neurais que realizam algum tipo de classificação, após as camadas de *pooling* e convolução, é adicionada pelo menos uma camada de neurônios totalmente conectados, responsável por realizar o caminho de decisão após receber como entrada as saídas advindas das camadas anteriores (VARGAS; PAES; VASCONCELOS, 2016).

Figura 4 – Rede neural convolucional e suas diferentes camadas.



Fonte: (VARGAS; PAES; VASCONCELOS, 2016)

Os detectores de objetos modernos, baseados em redes neurais convolucionais, podem ser divididos em dois grupos (LOHIA et al., 2021). O primeiro consiste em detectores em dois estágios, nos quais existe um estágio preliminar responsável pela extração de regiões de interesses, para que subsequentemente, sejam realizadas detecções nessas regiões. Já os detectores em um estágio, permitem a realização de detecções sem a necessidade de um estágio preliminar, tornando-os mais rápidos e permitindo, assim, uso em tempo real. Na Subseção 2.1.3.3.1 a serão abordados alguns dos principais algoritmos de detecção de objeto em dois estágios, enquanto na Subseção 2.1.3.3.2 serão abordados algoritmos de detecção de objetos em um estágio.

2.1.3.3.1 Detectores em dois estágios

Dentre os detectores de objetos em dois estágios, podemos destacar os seguintes:

- R-CNN: o funcionamento do R-CNN (GIRSHICK et al., 2014) se inicia com a extração de um conjunto de candidatos através da técnica de busca seletiva (SANDE

et al., 2011), após esta etapa, cada um dos candidatos é redimensionado a uma imagem de tamanho fixo que será processado em um modelo de redes neurais convolucionais treinado para extração de características. Por fim, através de um classificador linear SVM (*Support Vector Machine*), são realizadas detecções de objetos em cada uma das regiões candidatas. A técnica, entretanto, apresenta limitações de desempenho, levando em torno de 13 segundos para realizar a detecção em um único objeto (ZOU et al., 2019).

- SPPNet: os modelos CNN vistos até o surgimento do SPPNet (*Spatial Pyramid Pooling Networks*) (HE et al., 2015), exigiam imagens de tamanho fixo como entrada (244x244), deste modo, o SPPNet tem como principal contribuição a introdução da *Spatial Pyramid Pooling (SPP)*, uma camada extra que permite a rede neural utilize representações de tamanho fixo, independentemente do tamanho das imagens de entrada ou regiões de interesse, eliminando a necessidade de redimensionar imagens. Essa melhoria permite ao SPPNet um tempo de execução 20 vezes mais rápido quando comparado ao R-CNN, sem comprometer a acurácia das detecções (ZOU et al., 2019).

- Fast R-CNN: o detector Fast R-CNN (GIRSHICK, 2015) surgiu com melhorias das técnicas apresentadas por R-CNN (GIRSHICK et al., 2014) e SPPNet (HE et al., 2015), sendo capaz de realizar simultaneamente o treinamento de um detector e a regressão de caixas delimitadoras (LEE; KWAK; CHO, 2018). O Fast R-CNN apresentou um aumento de 58.5% para 70.0% no valor de mAP (*Mean Average Precision*) quando comparado com o R-CNN, além de realizar detecções 200 vezes mais rápido. Entretanto, o tempo de detecção ainda é limitado pela sua extração de regiões de interesse (ZOU et al., 2019).

- Faster R-CNN: o Faster R-CNN (REN et al., 2015) surge pouco após seu antecessor, o Fast R-CNN (GIRSHICK, 2015) e foi o primeiro detector de objetos fim-a-fim, não fazendo uso de componentes não-diferenciáveis, como o NMS (*Non-Maximum Supression*) (NEUBECK; GOOL, 2006), responsável pela remoção de predições redundantes (SUN et al., 2021). Também é considerado primeiro a obter desempenho próximo a tempo real, levando em torno de 200ms para realizar uma única detecção, 122ms a menos em relação ao seu antecessor (ZOU et al., 2019). Sua principal contribuição foi a RPN (*Region Proposal Network*) que permite a extração de candidatos mais rápida (10ms). Apesar de reduzir os gargalos observados no Fast R-CNN, o Faster R-CNN apresenta redundância de operações em estágios de detecção subsequentes, resultando em um tempo maior de treinamento (ZOU et al., 2019).

2.1.3.3.2 Detectores em único estágio

Dentre os detectores de objetos em um estágio, podemos destacar os seguintes:

- YOLO (You Only Look Once): enquanto os detectores citados nas subsecções anteriores utilizam uma abordagem em duas etapas (identificação de candidatos e verificação), no YOLO (Redmon et al., 2016), uma única rede neural é aplicada a uma imagem completa. A rede neural divide a imagem em regiões realizando predições de caixas delimitadora para cada uma das regiões simultaneamente, abordagem que torna o algoritmo extremamente rápido, chegando a taxas de 155fps (ZOU et al., 2019). O autor apresentou uma serie de melhorias em trabalhos subsequentes com a versão 2 (REDMON; FARHADI, 2016), versão 3 (REDMON; FARHADI, 2018) e a versão 4 (BOCHKOVSKIY; WANG; LIAO, 2020), utilizada neste trabalho. A principal limitação do YOLO quando comparado com detectores em dois estágios, é a perda de acurácia ao determinar a localização dos objetos, em especial, objetos pequenos contidos em cenas amplas.
- SSD: assim como as versões subsequentes do YOLO (REDMON; FARHADI, 2016; REDMON; FARHADI, 2018; BOCHKOVSKIY; WANG; LIAO, 2020), o SSD (*Single-Shot Multibox Detector*) (LIU et al., 2016) surge com o objetivo de aprimorar a acurácia dos detectores em uma etapa. As principais contribuições do SSD são as técnicas de detecção multi-referência e multi-resolução que trazem ganhos significativos de acurácia para objetos pequenos. Sua principal diferença em relação aos detectores citados anteriormente é a sua capacidade de realizar detecções de objetos em diferentes escalas através de diferentes camadas da rede neural, enquanto as demais técnicas realizam detecções apenas em suas camadas superiores (ZOU et al., 2019).

2.2 Sistemas Distribuídos

Segundo (STEEN; TANENBAUM, 2017), um sistema distribuído é uma coleção de elementos computacionais autônomos que se apresentam para o usuário como um sistema único e coerente.

Já (COULOURIS; DOLLIMORE; KINDBERG, 2005), define como sistemas distribuídos todos aqueles em que seus componentes de *hardware* e *software* conectados através de uma rede de computadores são capazes de se comunicar e coordenar suas ações através do envio de mensagens.

2.2.1 Metas de sistemas distribuídos

Ao se desenvolver um sistema distribuído, 4 metas são consideradas enquanto se avalia o esforço de implementação (STEEN; TANENBAUM, 2017), estas metas são:

- Compartilhamento de recursos: um sistema distribuído deve prover fácil acesso de recursos remotamente aos usuários.
- Transparência: o fato de que os recursos estão distribuídos através de múltiplas plataformas deve ser imperceptível ao usuário.
- Abertura: um sistema distribuído deve ser capaz de prover fácil acesso e integração de recursos com outras plataformas.
- Escalabilidade: um sistema distribuído deve ser capaz de crescer em número de usuários ou recursos, sem que isto afete seu desempenho de maneira significativa.

2.2.2 Modelos arquiteturais

A arquitetura de um sistema é a forma em que seus componentes estão organizados e como estes se relacionam (COULOURIS; DOLLIMORE; KINDBERG, 2005). Nesta subsecção serão abordados os principais modelos arquiteturais utilizados em sistemas distribuídos.

2.2.2.1 Arquitetura cliente-servidor

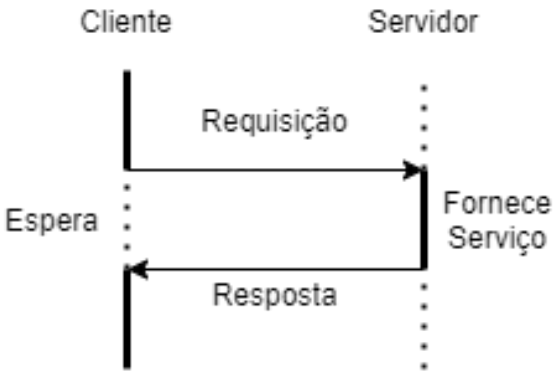
Na arquitetura cliente-servidor, os processos podem ser divididos em dois grupos. Os servidores podem ser vistos como processos que implementam algum tipo de serviço, como um banco de dados ou sistema de arquivos. Um cliente por sua vez, é um serviço que solicita um serviço fornecido por um servidor através do envio de uma requisição e de maneira subsequente, aguarda por uma resposta do servidor (STEEN; TANENBAUM, 2017). A Figura 5 ilustra como ocorre a comunicação entre um cliente e um servidor.

Neste tipo de arquitetura os servidores também podem atuar como clientes de outros servidores, como ocorre com serviços web que comumente podem atuar como clientes de um serviço de arquivos local que gerencia os arquivos onde a página web está hospedada (COULOURIS; DOLLIMORE; KINDBERG, 2005).

2.2.2.2 Arquitetura ponto-a-ponto

Na arquitetura ponto-a-ponto, comumente chamada de P2P, todos os processos envolvidos em uma tarefa desempenham papéis similares, cooperando como pontos sem distinção entre clientes e servidores, ou seja, todos os pontos, ou nós, executam

Figura 5 – Interação entre cliente e servidor.



Fonte: Adaptado de (STEEN; TANENBAUM, 2017)

uma mesma aplicação e oferecem o mesmo conjunto de interfaces aos outros nós (COULOURIS; DOLLIMORE; KINDBERG, 2005).

Em outras palavras, a arquitetura ponto-a-ponto pode ser definida como uma serie de sistemas e aplicações que empregam recursos distribuídos a fim de realizar uma tarefa de maneira descentralizada (MILOJICIC et al., 2002).

2.3 Avaliação de desempenho

Esta seção está dividida em duas partes: a Subseção 2.3.1 apresenta as principais métricas utilizadas para avaliação de desempenho de modelos destinados à detecção de objetos e a Subseção 2.3.2 aborda métricas e abordagens voltadas à avaliação de desempenho de serviços como API's (*Application Programing Interfaces*).

2.3.1 Métricas para detectores de objetos

A métrica mais difundida para a detecção de objetos em imagens é o AP (*Average Precision*) e suas variações (PADILLA; NETTO; SILVA, 2020). Esta subseção abordará os aspectos básicos desta métrica, bem como suas principais variações.

2.3.1.1 Verdadeiro positivo

True positive (TP) ou verdadeiro positivo, representa toda e qualquer detecção realizada de maneira correta (PADILLA; NETTO; SILVA, 2020).

2.3.1.2 Falso positivo

False positive (FP) ou falso positivo, representa toda e qualquer detecção realizada de maneira incorreta, por exemplo: detecção de um objeto inexistente; classificação ou posição incorreta do objeto detectado (PADILLA; NETTO; SILVA, 2020).

2.3.1.3 Falso negativo

False negative (FN) ou falso negativo, representa toda e qualquer não-detecção onde é esperada a detecção de algum objeto.

É importante salientar que verdadeiros negativos, ou *True Negatives* (TN) não se aplicam à detecção de objetos, uma vez que em uma única imagem podem haver inúmeros objetos que não devem ser detectados pelo modelo (PADILLA; NETTO; SILVA, 2020).

2.3.1.4 *Intersection over union*

Uma vez conhecidos os conceitos apresentados nas Subseções 2.3.1.1, 2.3.1.2 e 2.3.1.3, é necessário estabelecer o que é uma detecção correta e o que é uma detecção incorreta. Uma forma comum de definir estes aspectos é através da utilização de IOU (*Intersection over union*), métrica baseada no Coeficiente de Jaccard (JACCARD, 1901) que mede a similaridade entre dois conjuntos de dados.

No escopo da detecção de objetos, o IOU representa a área de sobreposição da caixa delimitadora detectada B_p e a caixa delimitadora da verdade de campo (*Ground truth*) B_{gt}, dividido pela área obtida pela união entre as duas caixas delimitadoras, o que pode ser expresso como:

$$IoU = \frac{area(B_p \cap B_{g_t})}{area(B_p \cup B_{g_t})}$$

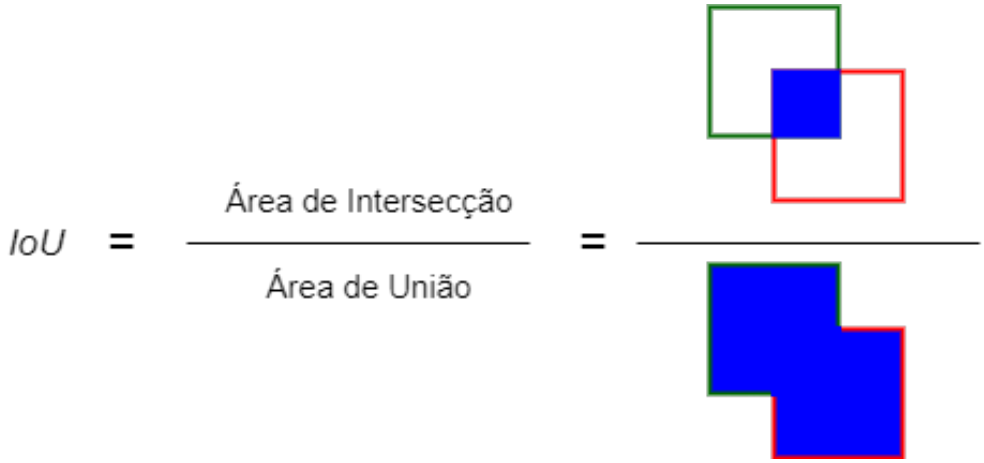
O IOU assume valores em um intervalo de 0 a 1, onde o 0 indica que não há sobreposição de caixas limitadoras e o 1 indica uma sobreposição perfeita. Juntamente com o IOU, é necessário definir um limiar α, no intervalo entre 0 e 1, que será usado para determinar se uma detecção será considerada correta ou incorreta (KOECH, 2020). Se $\alpha \leq IoU$, então a detecção será considerada correta, caso contrário, a detecção será considerada incorreta (PADILLA; NETTO; SILVA, 2020). A Figura 6 mostra uma representação visual do IOU.

Como dito na Subseção 2.3.1.3, verdadeiros negativos não são utilizados, desta forma, o uso de métricas baseadas em verdadeiros negativos é evitado, optando-se por métricas como Precisão e *Recall* (PADILLA; NETTO; SILVA, 2020), vistas na Subseção 2.3.1.5.

2.3.1.5 Precisão e Recall

Precisão é a capacidade de um classificador de identificar apenas objetos relevantes. Em outras palavras, precisão é a proporção de verdadeiros positivos (PADILLA; NETTO; SILVA, 2020).

Figura 6 – Representação visual do IOU



Fonte: Adaptado de (KOECH, 2020)

A precisão pode ser expressa matematicamente da seguinte forma:

$$P = \frac{TP}{TP + FP} = \frac{TP}{Total}$$

Recall, por sua vez, mede a capacidade do classificador encontrar todos os casos relevantes (ou seja, todas as *ground truths*). Em outras palavras, é a proporção de verdadeiros positivos detectados entre as verdades de campo ou *ground truths* (PADILLA; NETTO; SILVA, 2020).

O *recall* pode ser expresso matematicamente da seguinte forma:

$$R = \frac{TP}{TP + FN} = \frac{TP}{Ground\ truth\ total}$$

Um classificador é considerado eficiente quando capaz de identificar muitas verdades de campo (alto *recall*), ao mesmo tempo que consegue identificar objetos relevantes com frequência (alta Precisão). Um classificador perfeito, seria aquele com $FN = 0(R = 1)$ e $FP = 0(P = 1)$ (KOECH, 2020).

2.3.1.6 Curva PR

A curva de precisão-recall, ou curva PR, consiste no gráfico da precisão como uma função de *recall*. Este gráfico demonstra o equilíbrio entre as duas métricas variando o grau de confiabilidade do classificador. Se o número de falsos positivos é baixo, a precisão é alta, porém, mais instancias de objetos podem não ser detectadas, ocasionando um alto valor de falsos negativos (baixo *recall*). Por outro lado, se mais verdadeiros positivos forem considerados através da redução do limiar de IOU, o *recall* irá aumentar, porém mais falsos positivos podem ser detectados, diminuindo a precisão do classificador. Idealmente, um bom classificador deve manter altos níveis de precisão e *recall*, mesmo que o limiar de IOU varie (KOECH, 2020).

2.3.1.7 Average Precision

Average Precision, ou precisão média é, idealmente, a área abaixo da curva PR (KOECH, 2020), sendo matematicamente representada da seguinte forma:

$$AP@\alpha = \int_0^1 P(r)\,dr$$

Um alto valor de área sob a curva índica altos níveis de precisão e *recall*. Em casos práticos, a curva PR apresenta um comportamento em ”zigue-zague”, o que dificulta o cálculo da área sob a curva. Para que se obtenha um comportamento mais linear, são utilizadas duas abordagens: interpolação de 11 pontos e interpolação de todos os pontos (PADILLA; NETTO; SILVA, 2020).

Na interpolação de 11 pontos a curva PR é sumarizada através da média dos valores máximos de precisão em um conjunto de 11 níveis de *recall* igualmente espaçados [0, 0.1, 0.2, ..., 1], expresso por

$$AP@\alpha_{11} = \frac{1}{11}\sum_{r\in R} Pinterp(r),$$

onde $R = 0, 0.1, 0.2, ..., 1$ e

$$Pinterp(r) = \max_{r':r'\geq r} p(r')$$

Na interpolação de todos os pontos, em vez dos 11 pontos distribuídos anteriormente, a interpolação é realizada em todos os pontos, desta forma:

$$AP@\alpha_{all} = \sum_n (R_{n-1} - R_n) Pinterp(R_{n-1}),$$

onde

$$Pinterp(R_{n-1}) = \max_{R':R'\geq R_{n+1}} p(R')$$

2.3.1.8 Mean Average Precision

O *Mean average precision (mAP)* é uma métrica utilizada para se obter a precisão média de um classificador sobre todas as classe em uma base de dados particular. O mAP nada mais é do que a média das AP de todas as classes (PADILLA; NETTO; SILVA, 2020), sendo

$$mAP = \frac{1}{N}\sum_{i=0}(N)AP_i,$$

onde N é o número total de classes e AP_i é o AP da i-ésima classe.

2.3.1.9 F1-Score

Também conhecido como *F-Mesure* ou *F-Score*, o *F1-Score* é uma métrica que combina precisão e *recall* de um modelo[1] sendo definido como a média harmônica entre precisão P e recall R (SASAKI, 2007). A métrica é expressa matematicamente como:

$$F_1 = \frac{2PR}{P + R}$$

2.3.2 Avaliação de desempenho de API's

Nesta subseção serão introduzidos conceitos relacionados a testes de desempenho de API's, dentre eles, tipos de testes de performance e métricas utilizadas.

2.3.2.1 Testes de *baseline*

O objetivo de um teste de *baseline* é avaliar o comportamento de um determinado sistema sob um volume de normal de requisições, dentro de suas capacidades. Os resultados desse teste devem ser analisados a partir da média e do pico de tempo de resposta e taxa de erros. A utilização de CPU e memória por parte da aplicação também deve ser analisado para que se possam eliminar gargalos (DE, 2017).

2.3.2.2 Testes de carga

Em um teste de carga, o volume de requisições deve ser incrementado gradativamente ao longo do teste. O objetivo deste tipo de teste é ajudar a entender o comportamento esperado do sistema e sua capacidade de resposta a picos de carga. Métricas de performance do servidor, como taxa de uso da CPU e utilização de memória devem ser analisadas, a fim de compreender o estado da plataforma (DE, 2017).

2.3.2.3 Testes de stress

Em testes de stress, o objetivo é encontrar um ponto de falha da plataforma, ou seja, avaliar qual a capacidade máxima de uma plataforma. Neste tipo de teste, o tráfego é incrementado até que ocorram falhas ou erros em chamadas de API comecem a ocorrer com maior frequência (DE, 2017).

2.3.2.4 Testes de imersão

Em testes de imersão são aplicados volumes considerados normais em uma API durante um período prolongado de tempo. Seu objetivo é avaliar possíveis defeitos com a disponibilização de novos recursos e torna-los disponíveis no ciclo de execução seguinte. Se os recursos de um sistema não são disponibilizados periodicamente,

[1] <https://deepai.org/machine-learning-glossary-and-terms/f-score>

aumenta-se a probabilidade de o sistema apresentar falhas sob altas cargas (DE, 2017).

2.3.2.5 Métricas de desempenho de API

Para uma avaliação adequada do desempenho de uma API, as seguintes métricas devem ser avaliadas (DE, 2017):

- Tempo de resposta: métrica usada para calcular o tempo decorrido entre a solicitação de um recurso computacional e o completo atendimento da mesma solicitação. Tempo mínimo, mediana e máximo também devem ser avaliados.
- Latência: métrica usada para medir a latência ou atraso introduzido por algum fator intermediário, como um *gateway* usado na arquitetura da API.
- Vazão: mede o número de requisições atendidas em um determinado intervalo de tempo.
- Taxa de sucesso e erro: mede o número de requisições atendidas com sucesso ou o número de requisição atendidas com erros durante a carga.
- Uso de CPU: métrica utilizada para mensurar o uso corrente do poder de processamento de um computador ou a quantidade de tarefas simultâneas em processamento. O baixo uso de CPU indica que o sistema é capaz de atender um volume maior de tarefas, o contrário, indica que o sistema está sob stress.
- Utilização de memória *heap*: indica a quantidade de memória demandada pelo sistema durante a carga.

3 Trabalhos Correlatos

Neste capítulo serão apresentadas pesquisas que contribuem para a compreensão e comparação deste trabalho. A seção está dividida em duas partes, onde a primeira dá maior ênfase em pesquisas voltadas a implementação de modelos de aprendizagem profunda na detecção de armamentos. A segunda subseção, por sua vez, aborda pesquisas relacionadas a arquiteturas distribuídas para detecção de armas ou outros objetos.

3.1 Detecção de armas em tempo real

(OLMOS; TABIK; HERRERA, 2018) propuseram um sistema de alarme baseado em detecção de armas de fogo em imagens de video. O sistema em questão utiliza um modelo de aprendizagem profunda baseado em redes neurais convolucionais comparando os resultados em diferentes bases de dados com abordagens de janela deslizante e RPN, utilizando o algoritmo Faster R-CNN. Os autores obtiveram uma redução no número de falsos positivos ao aumentar o número de classes de duas (arma, não-armas) para 28 classes, distribuídas em diversos objetos diferentes, porém o tempo para processar uma única imagem com a abordagem de janela deslizante é de cerca de 1,5 segundos, deste modo, não se mostrou o modelo mais adequado para um sistema de detecção em tempo real. A abordagem utilizando RPN, entretanto, apresentou resultados mais promissores, tendo como tempo médio de detecção 0,19 segundos.

(GONZáLEZ et al., 2020) contextualizaram os maiores desafios que estão atrelados à detecção de objetos pequenos em imagens amplas. O grupo criou um conjunto de dados que combinava conjuntos de imagens já existentes, imagens anotadas manualmente após uma encenação de ataque a mãos armadas e um terceiro conjunto de imagens composto por cenas geradas graficamente a partir de um motor gráfico de jogos digitais. Diferente deste trabalho, os autores utilizaram Faster R-CNN para obter suas detecções. Foi observado que o uso de conjuntos de dados sintéticos contribuiu para melhores detecções, porém o tempo de resposta (0,90 segundos) não se demonstrou satisfatório para um contexto de detecção em tempo real.

(WARSI et al., 2019) apresentou um sistema de detecção de armas de fogo utilizando aprendizagem profunda, primariamente com o algoritmo Yolo em sua terceira versão. Para fins de treino, os autores criaram um dataset próprio e mesclaram com datasets colhidos do ImageNet (DENG et al., 2009). Os resultados apresentados são comparados aos resultados obtidos por (OLMOS; TABIK; HERRERA, 2018), desta

forma, o grupo concluiu que ambos os algoritmos, YOLOV3 e o Faster R-CNN apresentado por (OLMOS; TABIK; HERRERA, 2018) apresentam resultados satisfatórios, porém o YoloV3 se mostra mais vantajoso em termos de tempo de execução, sendo capaz de processar 45 frames por segundo, contra 8 frames por segundo do Faster R-CNN.

(GALAB; TAHA; ZAYED, 2020) propõe uma abordagem para detecção de armas de fogo através de imagens de circuito fechado. Isto é realizado através de redes neurais convolucionais, utilizando duas técnicas diferentes de aprendizagem profunda: AlexNet e GoogleLeNet. Ambas as técnicas apresentaram altos índices de acurácia, mesmo quando aplicado em imagens de baixa qualidade. Entretanto, os autores não indicaram o tempo de resposta para cada imagem obtida.

(RUIZ-SANTAQUITERIA et al., 2020) desenvolveram uma abordagem onde a detecção é realizada observando primeiramente a posição dos ombros e braços de um indivíduo em uma imagem, a fim de identificar uma posição de ameaça (apontando a arma na horizontal), seguida da análise de região da mão após a primeira parte da análise. Os resultados foram comparados com outras três abordagens, incluindo o algoritmo YOLO (Redmon et al., 2016) em sua terceira versão. A abordagem dos autores apresentou melhor precisão em relação as demais, porém não foram especificados os tempos, portanto não é possível afirmar se a abordagem é aplicável em um contexto de detecção em tempo real.

(NOOR; ISA et al., 2021) propuseram um arcabouço para detecção de armamentos utilizando-se do YOLOv4 (BOCHKOVSKIY; WANG; LIAO, 2020) para realizar esta tarefa através da análise frame a frame de imagens de circuito fechado. Os autores inicialmente propuseram um modelo com uma única classe genérica para representar diferentes tipos de armamento. Entretanto esta abordagem apresentou baixos valores de mAP após o treinamento devido a grande variedade de armamentos (rifles, pistolas, espadas e entre outros). Por fim, os autores adotaram uma abordagem com classes mais específicas (Handgun e Knife), obtendo melhores detecções, embora o modelo ainda apresente dificuldades em detectar objetos se movendo rapidamente ou em ângulos e posições diferentes das imagens de treino. Os autores, porém, não destacaram o desempenho dos modelos em função do tempo de execução.

Os trabalhos citados focam em diferentes abordagens para detecção de armas de fogo em tempo real. Este trabalho, por sua vez, visa identificar a viabilidade desse tipo de sistema em ambiente real. Por demonstrar melhores tempos de execução, este trabalho utilizará o algoritmo YOLO como prova de conceito, em sua versão mais recente (BOCHKOVSKIY; WANG; LIAO, 2020), pois apresenta maior precisão e taxa de frames por segundo.

3.2 Arquiteturas distribuídas

(DIEGO et al., 2019) desenvolveram e testaram em seu trabalho uma arquitetura distribuída para um sistema de monitoramento inteligente, capaz de disparar alarmes e de tratar diversos cenários diferentes simultaneamente através do uso de diversos sensores interconectados. Esta arquitetura foi construída utilizando o ROS (Robot Operating System), que permite o desenvolvimento e a conexão de diversos módulos via protocolo TCP-IP sem fio. A solução é capaz de monitorar e detectar pessoas e veículos em ambientes reais, sejam eles internos ou externos. A natureza distribuída da solução permite que cada um de seus módulos sejam executados em máquinas diferentes. Os resultados obtidos demonstram que cenários como invasão de áreas restritas, monitoramento de fluxo de pessoas, abandono de objetos e obstrução de câmeras são detectados de maneira confiável, sendo possível customizar o nível de segurança entre baixo, médio e alto para o disparo do alarme em cada cenário.

(Cheng; Sun; Tu, 2018) desenvolveram um arcabouço para facilitar a execução de redes neurais profundas em sistemas físicos com uma hierarquia de computação distribuída. Nesta proposta seria possível que modelos DDNN (*Distributed Deep Neural Networks*) fossem executados em dispositivos finais e servidores de maneira colaborativa com o auxílio de seus dois principais módulos: o módulo servidor, responsável pelo treinamento, geração de código em linguagem C dos modelos treinados e atender as requisições feitas a partir de dispositivos finais. O módulo de dispositivos, responsável por invocar inferências do servidor de um determinado modelo desejado. Os autores utilizaram como prova de conceito um sistema de monitoramento que verifica a entrada de animais em uma área restrita a pessoas, emulando as principais operações de uma rede neural convolucional destinada à detecção de objetos. As câmeras do experimento foram simuladas através do hardware Odroid-XU4.

(ROHIT, 2020) propôs um sistema de segurança para as redes de ônibus de Bangladesh que incluiria uma série de funcionalidades voltadas a detecção de incidentes, como incêndios, acidentes ou assédios e violência dentro dos coletivos. O autor desenvolveu um protótipo com arquitetura baseada em internet das coisas, onde é utilizado como hardware da aplicação, ligado aos principais sensores, o NodeMCU: placa utilizada para prototipação e desenvolvimento em IoT. Também se utilizou de um Raspberry Pi conectando-se a duas câmeras utilizadas para detecção de armas de fogo, armas brancas e incêndios. Os dados coletados a partir dos sensores e câmeras são passados para um *dashboard* online através de um broker MQTT. A solução proposta se mostrou eficiente em casos de incêndios e em detecção de sonolência, porém, o autor ressalta que as funcionalidades voltadas a detecção de armamentos necessitam de melhorias drásticas.

(HERRERA et al., 2018) propõe a divisão do pipeline utilizado pelo R-CNN,

utilizado como prova de conceito, em componentes discretos, tornando possível a distribuição desses componentes, aproveitando o poder de processamento de diferentes dispositivos. Desta forma, seria possível realizar a detecção de veículos em vagas de estacionamentos através de imagens sem a necessidade de servidores com GPU dedicados a esta função. O componente de RPA (*Region Proposal Algorithm*) foi implementado utilizando o algoritmo de EdgeBoxes, e pode ser observados ganhos de performance na execução do modulo de RPA com a execução simultânea em mais de um dispositivo.

Os trabalhos citados propõem abordagens que utilizam a computação distribuída como forma de otimizar o tempo de execução de algoritmos de aprendizagem profunda, a partir da execução de módulos em diferentes dispositivos. Dentre eles, apenas o trabalho desenvolvido por (ROHIT, 2020) tinha o mesmo propósito que este trabalho, pois propôs uma solução de segurança para transporte urbano que inclui a detecção de armamentos em imagens de circuito fechado embora não tenha apresentado os mesmos resultados vistos em outras funcionalidades. O autor faz uso de hardwares dedicados para este propósito, utilizando o Raspberry Pi como NVR (Network Video Recorder) que se comunica comunica com as interfaces através de serviços web. Neste trabalho, por sua vez, todo o processamento será realizado em um servidor a partir do acesso de um serviço. Como contribuição, este trabalho visa identificar como realizar detecções de armamentos em tempo real de maneira escalável. O resumo dos trabalhos apresentados pode ser visto na Tabela 2 e Tabela 3.

Tabela 2 – Tabela de resumo dos trabalhos relacionados (parte 1)

Trabalho	**Técnicas**	**Tempo Real**	**Nº de Classes**
(OLMOS; TABIK; HERRERA, 2018)	Faster R-CNN	Sim	28
(GONZáLEZ et al., 2020)	Faster R-CNN	Não	1(Gun)
(WARSI et al., 2019)	YOLOv3	Sim	1(Gun)
(GALAB; TAHA; ZAYED, 2020)	AlexNet; GoogleLeNet	Não	2 (Positive; Negative)
(RUIZ-SANTAQUITERIA et al., 2020)	PGC1; PGC2; Velasco-Mata; YOLOv3	Não	2 (Carrying Handgun; Not Carrying Handgun)
(NOOR; ISA et al., 2021)	YOLOv4	Não	2 (Handgun; Knife)
(DIEGO et al., 2019)	Gaussian Mixture of Models; Kalman Filter	Sim	N/A
(Cheng; Sun; Tu, 2018)	N/A	Sim	N/A
(ROHIT, 2020)	Viola-Jones	Sim	N/A
(HERRERA et al., 2018)	R-CNN	Não	N/A
Trabalho Proposto	YOLOv4	Sim	2 (Gun; Knife)

Fonte: O próprio autor

Tabela 3 – Tabela de resumo dos trabalhos relacionados (parte 2)

Trabalho	**Objetivo**	**Segurança Urbana**
(OLMOS; TABIK; HERRERA, 2018)	Desempenho	Não
(GONZáLEZ et al., 2020)	Desempenho	Não
(WARSI et al., 2019)	Desempenho	Não
(GALAB; TAHA; ZAYED, 2020)	Desempenho	Não
(RUIZ-SANTAQUITERIA et al., 2020)	Desempenho	Não
(NOOR; ISA et al., 2021)	Desempenho	Não
(DIEGO et al., 2019)	Escalabilidade	Não
(Cheng; Sun; Tu, 2018)	Desempenho	Não
(ROHIT, 2020)	Desempenho; Escalabilidade	Sim
(HERRERA et al., 2018)	Desempenho	Não
Trabalho Proposto	Desempenho; Escalabilidade	Sim

Fonte: O próprio autor

4 Metodologia

A seguir são apresentados os artefatos necessários para a realização dos experimentos, bem como as métricas e técnicas utilizadas para avaliação de desempenho. Na Seção 4.1 é apresentado o conjunto de dados utilizados para a realização do treinamento e testes do algoritmo YOLOv4, sendo esta seção dividida em duas subseções: a Subseção 4.1.1 correspondendo ao conjunto de imagens e a Subseção 4.1.2 correspondendo as anotações e seu formato. A Seção 4.2 é também dividida em duas partes: a Subseção 4.2.2 que aborda a etapa de treinamento do modelo de aprendizagem profunda usado neste trabalho, bem como as ferramentas utilizadas no processo e a Subseção 4.2.3 que aborda a validação do modelo mencionado na subseção anterior. A Seção 4.3 apresenta detalhes sobre a organização do experimento, bem como as entidades que o compõem e as ferramentas utilizadas.

4.1 Conjunto de dados

Podemos dividir o conjunto de dados entre arquivos de imagens e anotações, ambas descritas a seguir.

4.1.1 Imagens

O conjunto de imagens utilizado neste trabalho é composto por duas classes de armamentos: Pistolas e Facas. Estas imagens foram obtidas através de diferentes fontes. As imagens da classe faca usadas no treinamento e testes foram extraídas através do Google Open Images (KUZNETSOVA et al., 2020).

Deste *dataset* foram obtidas, com o auxílio da ferramenta de código aberto OIDv4 Toolkit (VITTORIO, 2018), 610 imagens da classe Facas, já as imagens da classe Pistolas foram obtidas através de outras duas fontes (THUAN, 2020; KUMAR, 2020) e mais 500 imagens de facas fornecidas por (SHEKHAR, 2020) complementaram o conjunto de imagens. A partir destas três novas fontes foram obtidas 4500 imagens da classe Pistola e 500 imagens da classe Facas, somando um total de 5610 imagens.

A Tabela 4 apresenta a distribuição de imagens entre treinamento e testes de acordo com as suas classes.

Figura 7 – Exemplo de imagem da classe “Pistolas” que compõe o conjunto de dados.

Fonte: (THUAN, 2020)

Figura 8 – Exemplo de imagem da classe “Facas” que compõe o conjunto de dados.

Fonte: (KUZNETSOVA et al., 2020)

4.1.2 Anotações

Para cada imagem do conjunto de dados é necessário que exista um arquivo texto com informações sobre as classes dos objetos contidos em determinada imagem, assim como suas coordenadas. Cada objeto é representado por cinco valores: classe, *x*, *y*, largura e altura. As classes são representadas por números inteiros de 0 ao número de classes menos 1. Neste trabalho são utilizadas apenas duas classes, portanto os valores 0 e 1 representam as classes Armas e Facas, respectivamente. Os valores *x*

Tabela 4 – Distribuição das imagens no conjunto de dados

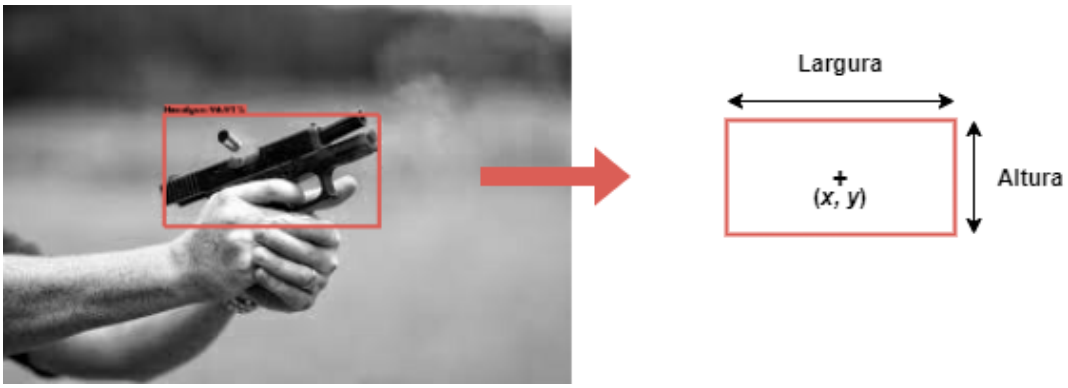
Classe	Total	Treino	Testes
Pistolas	4500	3000	1500
Facas	1110	813	297

Fonte: O próprio autor

e *y* representam as coordenadas do centro do objeto na imagem, podendo assumir valores normalizados entre 0 e 1, o que facilita na utilização de imagens em diferentes proporções. A largura e altura, assim como as coordenadas *x* e *y*, assume valores decimais normalizados entre 0 e 1, que são obtidos dividindo a altura e largura do objeto, em pixels, pela altura e largura total da imagem, também em pixels. O formato das anotações e exemplificado no Apêndice A.

O *dataset* fornecido por (SHEKHAR, 2020) não disponibiliza anotações junto com suas imagens, já o Google Open Images (KUZNETSOVA et al., 2020) fornece imagens com anotações no formato VOC Pascal (EVERINGHAM et al., 2010), diferente ao utilizado pelo algoritmo YOLO. Para o primeiro caso, as anotações foram feitas de forma manual com o auxilio da ferramenta de código aberto OpenLabeling (Cartucho; Ventura; Veloso, 2018). Já para o caso de anotações em diferentes formatos, foi realizada a conversão dos arquivos através de um script em Python[1], fornecido pelo OIDv4 Toolkit (VITTORIO, 2018).

Figura 9 – Coordenadas de um objeto contido em uma imagem.



Fonte: Adaptado de (KUMAR, 2020)

[1] <https://www.python.org/>

4.2 Modelo de Aprendizagem Profunda - YOLOv4

Esta seção aborda os modelos criados e está dividida em três partes: a Subseção 4.2.1 aborda detalhes do algoritmo de detecção de objetos YOLO. A Subseção 4.2.2 discorre sobre os aspectos técnicos do treinamento, enquanto a Subsecção 4.2.3 trata da avaliação dos modelos obtidos após treinamento do algoritmo.

4.2.1 You Only Look Once (YOLO)

O YOLO é um algoritmo de visão computacional capaz de detectar objetos em imagens ou vídeos (BOCHKOVSKIY; WANG; LIAO, 2020; Redmon et al., 2016). O YOLO consiste em uma rede neural convolucional capaz de processar uma imagem inteira, com uma única rede neural, sendo assim, suas predições são informadas pelo contexto global das imagens[2].

O algoritmo divide a imagem de entrada em *grid* ou malha *S x S*. Se o centro de um objeto estiver contido em uma célula da malha, esta célula será responsável pela identificação do objeto (Redmon et al., 2016).

Cada célula prediz um valor *B* de caixas delimitadoras e grau de confiabilidade para cada uma das caixas. Caso não haja nenhum objeto na célula, o grau de confiabilidade será 0, caso contrário, o grau de confiabilidade será igual a probabilidade para um determinado objeto multiplicado pelo *IOU* entre a caixa delimitadora da predição e a caixa delimitadora da verdade de campo (*Ground Truth*) (Redmon et al., 2016).

Cada caixa delimitadora é composta por 5 componentes (*x, y, h, w, confiabilidade*). As coordenadas *(x, y)* correspondem ao centro da caixa delimitadora, assumindo valores relativos à célula em que está localizada. Os valores *h* e *w*, correspondem à altura e largura, respectivamente. Por fim, a confiabilidade da predição representa o *IOU* entre a caixa delimitado e alguma verdade de campo (Redmon et al., 2016).

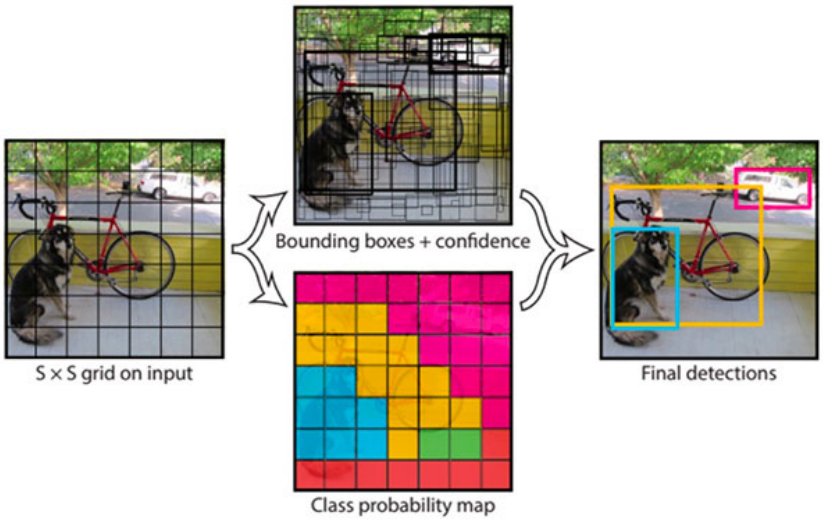
Também são realizadas, para cada célula da malha, predições condicionais de classes, expressos pela equação: $Pr(classe_i|Objeto)$. Esta probabilidade está condicionada às células da malha, onde um determinado objeto está contido. A rede prediz apenas um conjunto de probabilidade de classe por célula, independentemente do número de caixas delimitadoras *B*, o que representa um total de *S X S X C* probabilidades de classes, onde *C* é o número total de classes do modelo (Redmon et al., 2016).

4.2.2 Treinamento

O treinamento do modelo de aprendizagem profunda YOLOv4 foi realizado a partir das imagens e suas respectivas anotações coletadas para a composição do

[2] <https://viso.ai/deep-learning/yolov3-overview/>

Figura 10 – YOLO - Extração de caixas delimitadoras e probabilidade de classe.



Fonte: (Redmon et al., 2016)

conjunto de dados de treino, mostrados na Tabela 4. O treinamento foi realizado a partir do Darknet e a fim de acelerar o processo, foi utilizado um modelo pré-treinado fornecido pelos desenvolvedores(REDMON, 2013–2016). Este modelo é capaz de detectar 80 classes e foi treinado com o conjunto de dados COCO (*Commom Objects in Context*) (LIN et al., 2014). Para que fosse possível utilizar aceleração por GPU o treinamento foi realizado através de um notebook na plataforma Google Colaboratory[3], que permite ao usuário uso contínuo de um ambiente com GPU por 12 horas.

Ao realizar o treinamento através do Darknet, é necessário definir o número de lotes utilizados, número de iterações, número de classes e o número de filtros utilizados nas camadas de convolução da rede neural[4]. Estes parâmetros são definidos a partir de um arquivo de configuração utilizado pelo Darknet. O número de iterações definido neste arquivo corresponde a 2000 vezes o número de classes, para modelos com 4 classes ou mais, e igual a 6000 para modelos com até 3 classes. Um arquivo de classes, exemplificado no Apêndice B, também é utilizado para definir nomes para as classes do modelo.

4.2.3 Avaliação de desempenho de modelo

A acurácia do modelo de detecção obtido na etapa de treinamento foi medida a partir da execução do algoritmo sob o conjunto de dados de testes, especificados na Tabela 4, uma única vez. Tanto o modelo final, como os modelos parciais foram avaliados, a fim de identificar qual apresentava maiores taxas de precisão. O Darknet

[3] <https://colab.research.google.com/drive/1_GdoqCJWXsChrOiY8sZMr_zbr_fH-0Fg?usp=sharing>
[4] <https://github.com/AlexeyAB/darknet>

(REDMON, 2013–2016) também foi utilizado para esse propósito, pois o arcabouço permite a aferição de métricas como, precisão, *recall* e mAP, vistas na Subseção 2.3.1.

4.3 Experimento

O experimento visa simular, de maneira próxima a de um ambiente real, o envio de frames de vídeos captados por câmeras dispostas em coletivos. Para realização deste fim, o modelo gerado a partir do treinamento do YoloV4 foi convertido para um modelo compátivel com o framework Tensorflow[5], sendo possível realizar a hospedagem de uma API Rest[6], escrita em Phyton com o microframework Flask[7], responsável por atender as requisições recebidas. A hospedagem desta API foi feita na plataforma FloydHub[8], que dispõe de duas máquinas virtuais, que fornecem aceleração por GPU. A Tabela 5 apresenta as configurações de *hardware* da máquina utilizada no experimento.

Tabela 5 – Configurações de *hardware* no Floydhub.

GPU	Tesla V80 - 12GB de memória
CPU	Intel Xeon - 2 cores
Memória RAM	61GB
Espaço em disco	200GB SSD

Fonte: O próprio autor

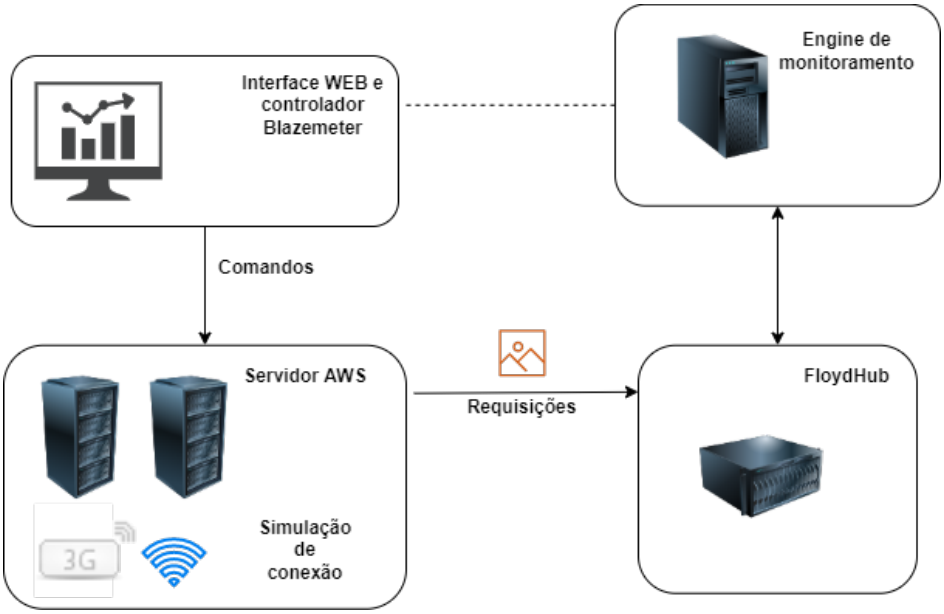
Para a simulação das câmeras de ônibus, foram criados grupos de usuários virtuais dentro da ferramenta Jmeter (HALILI, 2008). Cada um desses usuários virtuais envia requisições contendo uma imagem do conjunto de dados (teste) em seu corpo, de maneira simultânea. Para reduzir gargalos e problemas de desempenho introduzidos por limitações de recursos computacionais da máquina que hospeda o cliente Jmeter e consequentemente aumentar o número de usuários virtuais, o plano de testes criado no Jmeter foi exportado para a ferramenta web BlazeMeter[9].

O BlazeMeter é uma ferramenta capaz de realizar testes performance em aplicações Web e APIs, em seu plano gratuito permite a utilização de até 50 usuários virtuais, com execuções de até 20 minutos e fornece funcionalidades como a simulação da qualidade e do tipo de conexão utilizada, localização e realização de relatórios de execução. Os usuários virtuais criados no plano de testes, dentro do BlazeMeter, são

5 <https://www.tensorflow.org/?hl=pt-br>
6 <https://github.com/theAIGuysCode/Object-Detection-API>
7 <https://flask.palletsprojects.com/>
8 <https://www.floydhub.com/>
9 <https://www.blazemeter.com/>

executados por máquinas dos servidores AWS da Amazon. A Figura 11 mostra um esquema da organização do experimento.

Figura 11 – Infraestrutura do experimento realizado.



Fonte: O próprio autor

O plano gratuito da ferramenta, como mencionado anteriormente, apresenta algumas limitações de disponibilidade de recursos. Além do tempo máximo de execução e da quantidade de usuários virtuais, existem restrições para a simulação de requisições advindas de diferentes localidades e a disponibilidade dos relatórios gerados, é de uma semana. A Subseção 4.3.1 explana as configurações adotadas, dentro das limitações apresentadas pela ferramenta.

4.3.1 Configuração experimental do teste de carga

A fim de compreender o comportamento da API em um ambiente onde há um alto volume de câmeras realizando requisições, com possibilidade de ser escalado, as câmeras, representadas por grupos de usuários na ferramenta são introduzidos gradualmente durante a execução. Desta forma é possível analisar a partir de que número de usuários a API apresentará comportamento instável.

Neste experimento, o teste de carga se inicia com um número de 4 usuários virtuais. Durante os 15 minutos inciais (tempo de *ramp up*), novos usuários virtuais são inseridos de forma igualmente espaçada durante este intervalo, inserindo 4 novos usuários a cada 75 segundos. Cada grupo de 4 usuários virtuais, representa um ônibus. Durante seus 5 minutos finais o plano de testes é executado com sua carga máxima de 48 usuários virtuais, que representam 12 ônibus equipados com 4 câmeras cada.

Figura 12 – Distribuição das câmeras ao longo da execução.

Fonte: O próprio autor

O Blazemeter em sua versão com assinatura permite distribuir os usuários virtuais em servidores distribuídos em diferentes localidades. Em sua versão gratuita, porém, só é possível utilizar uma localidade. Deste modo, visando maior aproximação a um ambiente real, foram utilizados os servidores AWS da Amazon, todos localizados em Óregon (EUA), mesma localidade dos servidores do FloydHub que hospedam a API deste trabalho.

A emulação de rede, neste trabalho, atende o requisito mínimo de conexão (3G ou superior) exigido para a transmissão de dados em linhas de transportes coletivos[10]. A conexão 3G pode ser emulada em três níveis: pobre, médio e bom. Neste experimento, está sendo utilizada os níveis pobre e médio de conexão que apresentam por padrão na ferramenta uma largura de banda máxima por usuário de 780Kb e uma latência de 100 milissegundos.

A Tabela 6 apresenta os valores dos parâmetros configurados no plano de testes.

[10] <https://www.granderecife.pe.gov.br/sitegrctm/wp-content/uploads/2016/12/Manual-de-Operações-do-STTP-RMR-Anexo-16.pdf>

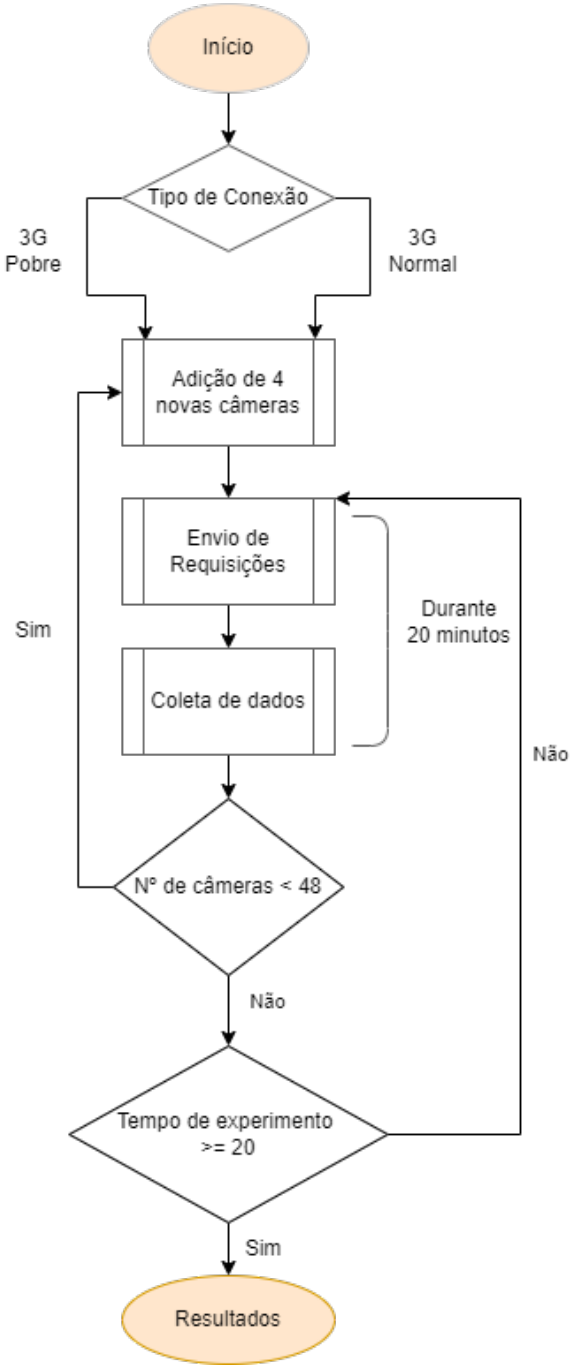
Tabela 6 – Configuração experimental do plano de testes

Total de Usuários	48
Duração	20 Minutos
Tempo de Ramp Up	15 Minutos
Localização	Óregon (EUA) - Amazon Web Services
Emulação de Rede	3G Pobre; 3G Médio
Largura de Banda/Usuário	780 Kb
Latência de Rede	100 ms

Fonte: O próprio autor

A Figura 13 apresenta o fluxo de execução do plano de testes na ferramenta. Após a execução, um relatório de execução foi gerado.

Figura 13 – Fluxo de execução do plano de testes.



Fonte: O próprio autor

5 Resultados e discussão

Neste capítulo são expostos e discutidos os resultados obtidos após a treinamento do modelo detecção, descrito na Seção 4.2, assim como os resultados obtidos após a execução do experimento descrito na Seção 4.3.

Este capítulo está dividido em duas seções: a Seção 5.1 destina-se a expor aos resultados obtidos a partir da inferência dos modelos de detecção de objetos, enquanto a Seção 5.2 expõe os resultados obtidos a partir dos testes realizados na API.

5.1 Avaliação de desempenho do modelo de detecção

A Tabela 7 apresenta os valores obtidos pelos modelos final e parciais após fim do treinamento. É possível observar que a classe Pistola, após a 2000ª interação, apresentou precisão média (AP) superior a 80% em todos os modelos parciais, tendo atingido seu valor ótimo na 3000ª iteração. A classe Faca, por sua vez, apresentou precisão média abaixo de 80% em todos os modelos parciais, atingindo seu valor ótimo na 4000ª iteração. Também se pode observar que o modelo obtido na 4000ª iteração apresentou menor número de falsos positivos (FP) para ambas as classes e maior número de verdadeiros positivos para a classe Faca.

Tabela 7 – Resultados da inferência por classe.

Nº de Iterações	Classe	AP	TP	FP
1000	Pistola	59.96%	1336	947
	Faca	13.67%	35	126
2000	Pistola	89.21%	1591	218
	Faca	67.38%	235	73
3000	Pistola	92.04%	1646	194
	Faca	74.67%	268	60
4000	Pistola	89.63%	1637	185
	Faca	77.30%	271	52
5000	Pistola	89.07%	1619	188
	Faca	73.01%	267	64
6000	Pistola	88.84%	1611	201
	Faca	72.52%	262	65

Fonte: O próprio autor

A partir da Tabela 8, é possível observar que o modelo obtido a partir da 4000ª iteração apresenta maior precisão, ou seja, maior percentual de classificações corretas. O mesmo ocorre com o mAP, que considerou apenas classificações onde o IOU (intersecção sobre união) se apresentou acima de 50%. Observa-se também que os modelos obtidos a partir da 3000ª e 4000ª iteração apresentam os menores números de falsos negativos (FN), ou seja, objetos que não foram identificados e classificados pelo modelo. Sendo assim, os dois modelos também apresentam maior *Recall*, Precisão e *F1-Score*, indicando melhor qualidade na detecção de verdadeiros positivos (TP).

Tabela 8 – Resultado da avaliação de desempenho do modelo.

Nº de iterações	**Precisão**	**Recall**	***F1-Score***	**FN**	**IoU Médio**	**mAP@0.5**
1000	0.56	0.64	0.60	783	39.85%	36.81%
2000	0.86	0.85	0.86	328	67.96%	78.29%
3000	0.88	0.89	0.89	240	71.43%	83.36%
4000	0.89	0.89	0.89	246	72.44%	83.46%
5000	0.88	0.88	0.88	268	71.69%	81.04%
6000	0.88	0.87	0.87	281	71.36%	80.68%

Fonte: O próprio autor

5.2 Avaliação de desempenho da API

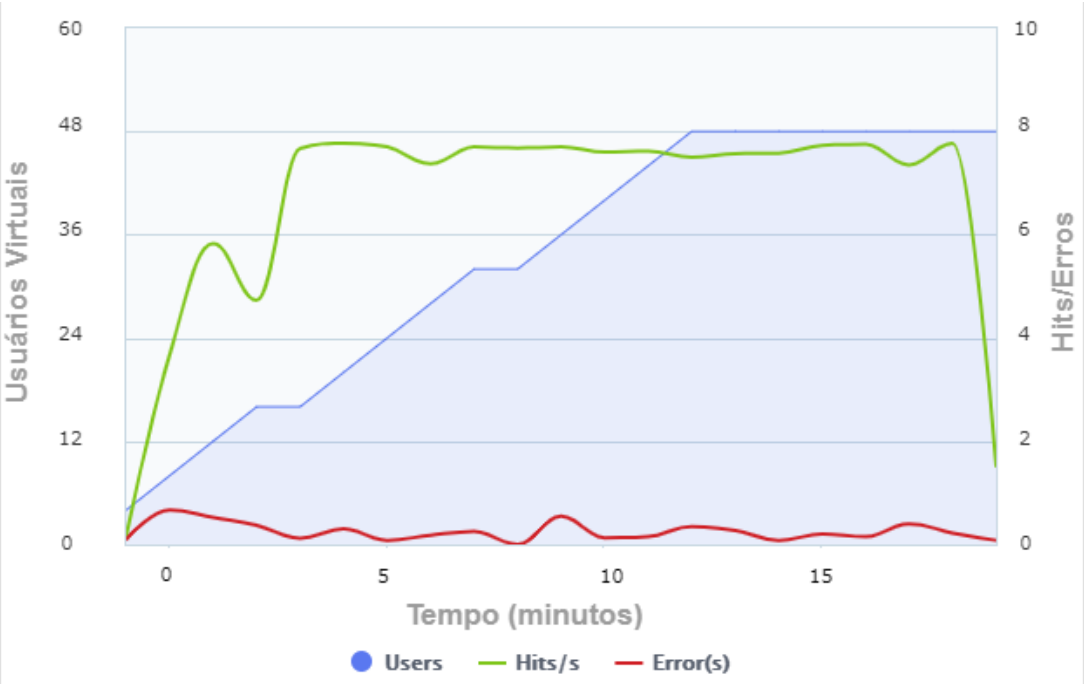
Nesta seção são apresentados os resultados, obtidos após realização do teste de carga descrito na Seção 4.3, em duas partes. Cada uma referente a um tipo de conexão utilizado durante a execução.

5.2.1 Simulação de conexão 3G pobre

A Figura 14 apresenta as taxa de requisições não atendidas (erros) ao longo do experimento, à medida em que são inseridos novos usuários virtuais, representados no gráfico pela curva em azul e pelo eixo vertical à esquerda. O eixo vertical à direita representa os volume de erros e a vazão em hits/s. É possível observar que a taxa de erros se manteve constante durante a execução, apresentando um percentual de 3.74%. Também observa-se que após a inserção de 16 usuários virtuais, a taxa de requisições por segundo (Hits/s) se mantém próxima a 7 Hits/s.

A Figura 15 apresenta o tempo de resposta ao longo do experimento. Após o primeiro passo do *ramp up*, o gráfico um aumento gradual a medida que novos usuários virtuais são inseridos. O tempo médio de reposta pra uma requisição neste cenário foi

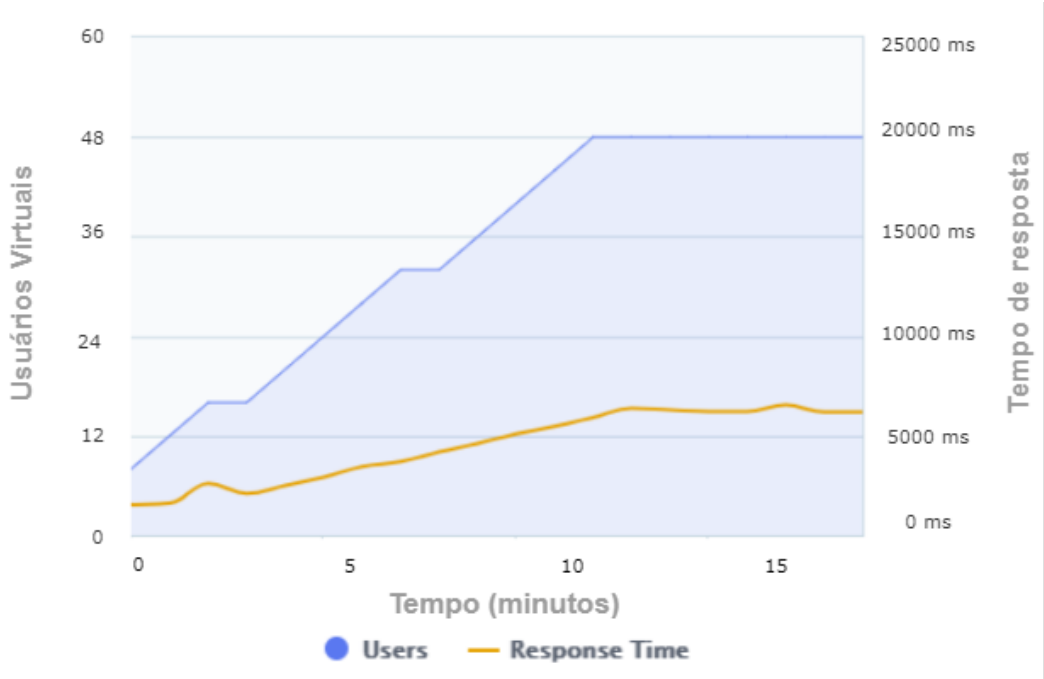
Figura 14 – Gráfico de requisições por segundo e erros verificados.



Fonte: O próprio autor

4.58 segundos e o menor tempo de resposta registrado durante essa execução foi de 437 milissegundos.

Figura 15 – Gráfico de tempo de resposta.

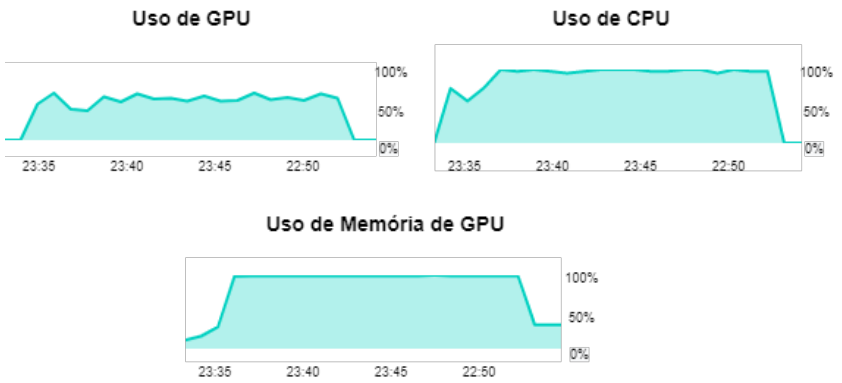


Fonte: O próprio autor

A taxa de uso de CPU no ambiente do FloydHub, com simulação de conexão 3G

pobre, se manteve acima de 90% a partir da inserção de 16 usuários virtuais(cada um representando uma única câmera), chegando a apresentar taxa igual a 100% ao atingir 28 usuários virtuais e novamente ao atingir 40 usuários virtuais. O uso de memória de GPU apresentou comportamento semelhante, onde após a inserção de 16 usuários virtuais, manteve uma taxa de 99.1% até o encerramento da execução. A taxa de uso de GPU, entretanto, no mesmo intervalo, não apresentou valores acima 64% de utilização, oscilando dentro de um intervalo entre 40% e 64% de ocupação. Os dados podem ser observados na Figura 16.

Figura 16 – Taxa de uso de CPU, GPU e memória de GPU no ambiente do FloydHub.



Fonte: O próprio autor

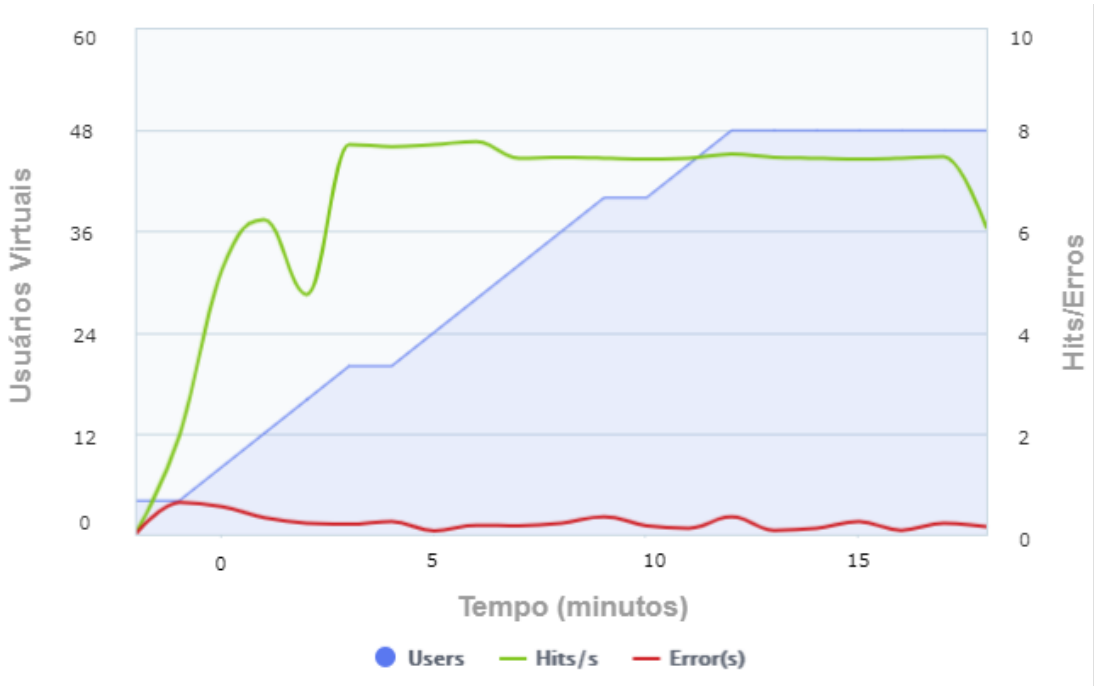
5.2.2 Simulação de conexão 3G médio

A execução realizada com configuração de conexão equivalente ao 3G de qualidade mediana apresentou desempenho similar à execução com 3G de qualidade inferior, apresentando uma incidência de erros inferior, com um taxa de erros de 3.45% de erros obtidos durante o envio de requisições. O comportamento pode ser ser observado na Figura 17.

O gráfico do tempo de resposta, ilustrado na Figura 18, apresentou comportamento similar ao apresentado na Subseção 5.2.1, com aumento gradual que acompanha o aumento do número de usuários virtuais. O tempo médio de resposta neste cenário, foi de 4.61 segundos e o tempo mínimo registrado foi de 452 milissegundos. Tempos ligeiramente mais altos do que os apresentados na Seção 5.2.1.

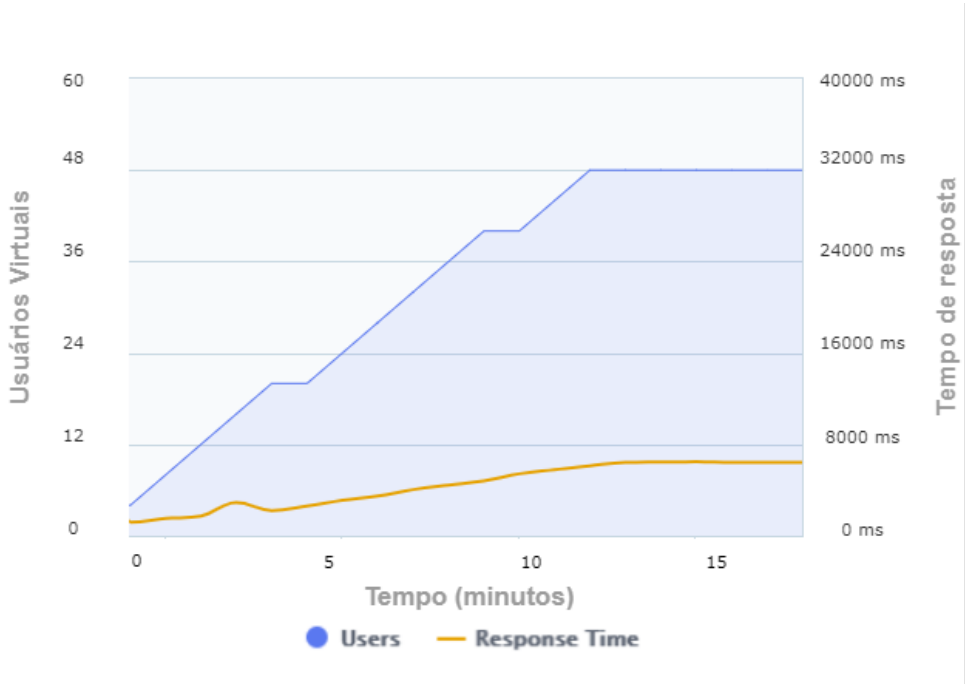
O taxa de uso CPU no ambiente do FloydHub se manteve acima de 90% durante quase toda a execução, chegando a apresentar taxa de 100% ao atingir 48 usuários virtuais. Comportamento semelhante ocorre com a taxa de utilização da memória da GPU, que se mantém em 99.1% a partir do momento em que o teste atinge 20 usuários virtuais, apresentando taxas de 100% ocasionalmente, após atingir 40 usuários. A taxa

Figura 17 – Gráfico de requisições por segundo e erros verificados.



Fonte: O próprio autor

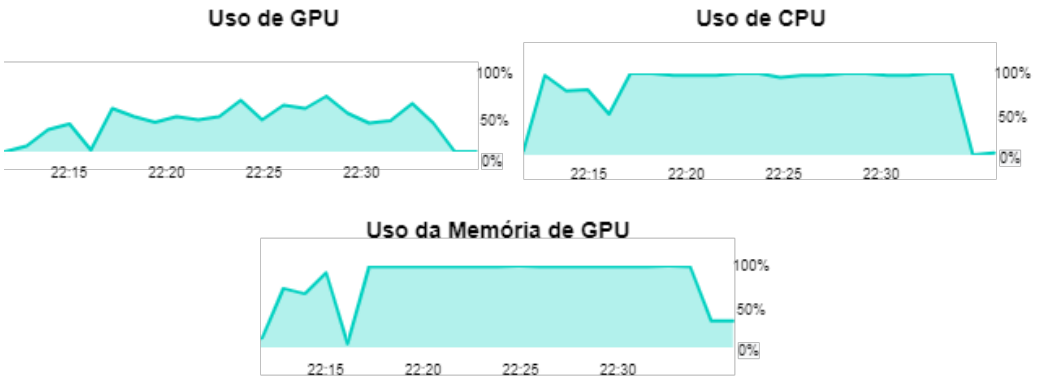
Figura 18 – Gráfico de tempo de resposta.



Fonte: O próprio autor

de uso da GPU, entretanto, atinge um pico de 68% quando o teste já apresentava 48 usuários virtuais. Estes dados podem ser visto na Figura 19.

Figura 19 – Taxa de uso de CPU, GPU e memória de GPU no ambiente do FloydHub.



Fonte: O próprio autor

A Tabela 9 apresenta um sumário das métricas colhidas pelo monitor da ferramenta Blazemeter, comparando os dados obtidos nas duas execuções realizadas.

Tabela 9 – Resumo das métricas colhidas após teste de carga.

Tipo de conexão	**Throughput médio**	**Taxa de erros**	**Tempo médio de resposta**	**Largura de banda média**
3G Pobre	6.90 hits/s	3.74%	4.58 s	373.78 KiB/s
3G Médio	6.86 hits/s	3.45%	4.61 s	373.01 KiB/s

Fonte: O próprio autor

5.3 Discussão

Neste trabalho foi utilizado o algoritmo YOLOv4 para realizar detecções de duas classes de objetos: Pistolas e Facas. O treinamento e validação do modelo foram realizados a partir da ferramenta de código aberto, Darknet. Subsequentemente, foi realizado um experimento onde o modelo foi implantado em um servidor utilizado para atender a carga de câmeras(ou clientes) que foram simuladas através da ferramenta de testes de carga, Blazemeter.

Dados os resultados apresentados nas seções anteriores, podemos observar, no modelo final após 6000 iterações e nos modelos parciais, que a precisão média para a classe Faca se mostrou inferior em relação a classe Pistola. Um dos possíveis fatores que contribuiu para a menor precisão da classe Faca, foi a distribuição de imagens utilizada no conjunto de dados. A menor precisão obtida pode estar associada também ao número consideravelmente menor de imagens da classe Faca utilizadas durante o treinamento, como mostrado na tabela 4.

Um segundo possível fator impactante na qualidade do detector é o viés de objeto (WANG; NARAYANAN; RUSSAKOVSKY, 2020) que pode ter sido introduzido pelo conjunto de dados de (SHEKHAR, 2020), devido a sua baixa variedade de posições, ângulos e formas. As imagens contidas neste conjunto, em sua maioria, exibem as facas na perspectiva do portador do objeto. A baixa variedade de posições também pode estar correlacionada com a quantidade de falsos positivos observada, uma vez que em contextos de violência urbana, ambas facas e armas de fogo são empunhadas de maneira similar (PÉREZ-HERNÁNDEZ et al., 2020).

Considerando o conteúdo das bases de dados usadas para compor o conjunto de dados deste trabalho, é importante ressaltar que a qualidade das imagens (superior as das câmeras de segurança) usadas tanto para treino como para validação do modelo, assim como suas cenas (poucas imagens em cenários amplos), podem ter contribuído para os valores de mAP apresentados nas seções anteriores, podendo assim, apresentar desempenho inferior com imagens de ambiente real.

Observando os tempos de resposta obtidos durante a execução do teste de carga, nota-se que o algoritmo é capaz de prover detecções próximas a tempo real, já que em ambas as execuções, o menor tempo observado, para que que uma única requisição fosse atendida, se apresentou abaixo de 0.5 segundos. Entretanto com a crescente demanda de detecções sendo realizadas simultaneamente, a API apresentou tempos de resposta cada vez mais altos ao longo da execução, atingindo médias de 4.58 segundos, para 3G pobre e 4.61 segundos para 3G médio, ao final do experimento, não conseguindo realizar detecções em tempo real sob altas demandas.

A partir dos resultados obtidos, observa-se que uma das possíveis limitações de desempenho da API, é a configuração de GPU. Após a inserção de 16 usuários virtuais a taxa de ocupação da memória da GPU permanece próxima a 100% durante o restante do experimento, embora a taxa de uso de GPU tenha atingido 68% apenas durante a carga máxima do experimento. A CPU também apresentou taxa de ocupação próxima a 100% durante toda a duração dos experimentos, fator que pode ter ocasionado a taxa de erros constante. Considerando estes dados, o uso de uma segunda GPU ou de uma única GPU com memória superior pode, potencialmente, reduzir o gargalo apresentado após a inserção de 16 usuários, provendo execuções mais próximas do tempo real. Um possível fator a ser considerado em trabalhos futuros, é o envio frame a frame das imagens de vídeo. Uma vez que houveram requisições não atendidas durante o experimento, o envio de frames em intervalos regulares pode reduzir o *throughput* médio e consequentemente reduzir o número de erros de requisições.

5.4 Considerações finais

Neste capítulo foram apresentados e discutidos os resultados do modelo de aprendizagem profunda após treinamento, bem como o seu desempenho após implantação em um servidor dedicado a atender requisições de clientes (câmeras). Por fim, serão descritas, no próximo capítulo, as conclusões, os trabalhos futuros e as limitações deste trabalho.

6 Conclusão

Em função das taxas de incidentes com armamentos dentro dos ônibus, este trabalho analisou a viabilidade de operação de um sistema de segurança destinado à detecção de armas de fogo e armas brancas em tempo real dentro de linhas de transporte público.

O experimento foi realizado visando simular um sistema de detecção de armamentos sob uma alta demanda de requisições de câmeras de ônibus, a fim de compreender sua viabilidade e limitações.

A partir dos resultados obtidos, observou-se que o modelo cliente-servidor utilizado sob as configurações mínimas oferecidas pelo FloydHub apresentou um tempo médio de reposta de 4.58 segundos, ao simular uma conexão 3G pobre, e de 4.61 segundos, ao simular uma conexão 3G mediana, tempo que não pode ser classificado como tempo real, dentro do contexto de segurança do transporte público, entretanto sendo possível sua implantação em outros contexto, como segurança residencial ou de estabelecimentos, considerando que a queda de desempenho observada ocorreu apenas a partir da inserção de 16 usuários virtuais(câmeras).

6.1 Limitações

As principais limitações deste trabalho são:

- Ausência da investigação de desempenho de outros algoritmos de aprendizagem profunda.
- Baixa variedade de configurações de *hardware* devido ao alto custo de utilização das plataformas.
- Conjunto de dados desbalanceado e apresentando viés na classe Facas.
- Ausência de imagens de resoluções inferiores e baixo número de imagens de cenas amplas durante o treinamento.

6.2 Trabalhos futuros

Dentre as sugestões de trabalhos futuros, podemos elencar:

- Criação de um novo conjunto de dados balanceado.

- Verificar o desempenho utilizando modelos arquiteturais como o P2P.
- Avaliar o desempenho de outros algoritmos de detecção de objetos em tempo real.
- Avaliar o desempenho em diferentes configurações de ambiente(GPU, memória e CPU).
- Executar simulações em ambiente real com câmeras IP.
- Desenvolvimento e teste de aplicativos móveis, como piloto, a serem utilizados por agentes de segurança em ambiente real.
- Investigar o desempenho de um detector de objetos em tempo real utilizando IoT.
- Mapear o ambiente de execução a fim de identificar gargalos e pontos de otimização, permitindo assim estimar a infraestrutura necessária.

Referências

AMIT, Y.; FELZENSZWALB, P.; GIRSHICK, R. Object detection. *Computer Vision: A Reference Guide*, Springer, p. 1–9, 2020. Citado na página 19.

BAHLMANN, C. et al. A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In: . [S.l.: s.n.], 2005. p. 255 – 260. ISBN 0-7803-8961-1. Citado na página 21.

BARBASCHOW, A. *NSW Police using artificial intelligence to analyse CCTV footage*. 2021. Disponível em: <https://www.zdnet.com/article/nsw-police-using-artificial-intelligence-to-analyse-cctv-footage/>. Citado na página 14.

BENGIO, Y.; GOODFELLOW, I.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press Massachusetts, USA:, 2017. v. 1. Citado na página 21.

BHANDARI, B.; PARK, G. Development of a real-time security management system for restricted access areas using computer vision and deep learning. *Journal of Transportation Safety & Security*, Taylor & Francis, p. 1–16, 2020. Citado na página 19.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. Citado 4 vezes nas páginas 15, 25, 34 e 42.

BRUIJN, D. D. et al. Human factor guidelines and workload in cctv design. In: . [S.l.: s.n.], 2015. Citado na página 14.

CAI, B. et al. Online exemplar-based fully convolutional network for aircraft detection in remote sensing images. *IEEE Geoscience and Remote Sensing Letters*, IEEE, v. 15, n. 7, p. 1095–1099, 2018. Citado na página 21.

CARDOSO, M. H. S.; SANTOS, T. F.; SILVA, M. A. V. d. Violence in public transport: An analysis of resilience and vulnerability in the city of rio de janeiro. *urbe. Revista Brasileira de Gestão Urbana*, SciELO Brasil, v. 13, 2021. Citado na página 14.

Cartucho, J.; Ventura, R.; Veloso, M. Robust object recognition through symbiotic deep learning in mobile robots. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.: s.n.], 2018. p. 2336–2341. Citado na página 41.

CHEN, J. *Neural Network*. 2020. Disponível em: <https://www.investopedia.com/terms/n/neuralnetwork.asp>. Acesso em: 30 jun. 2021. Citado na página 21.

Cheng, M.; Sun, Q.; Tu, C. An adaptive computation framework of distributed deep learning models for internet-of-things applications. In: *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. [S.l.: s.n.], 2018. p. 85–91. Citado 4 vezes nas páginas 15, 35, 37 e 38.

CNI, C. N. D. I. *Retratos da sociedade brasileira*. [S.l.]: CNI Brasília, 2018. Citado na página 13.

COULOURIS, G. F.; DOLLIMORE, J.; KINDBERG, T. *Distributed systems: concepts and design*. [S.l.]: pearson education, 2005. Citado 3 vezes nas páginas 25, 26 e 27.

CUI, L. et al. Edge learning for surveillance video uploading sharing in public transport systems. *IEEE Transactions on Intelligent Transportation Systems*, v. 22, n. 4, p. 2274–2285, 2021. Citado na página 15.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. [S.l.: s.n.], 2005. v. 1, p. 886–893 vol. 1. Citado na página 20.

Data Science Academy. *Deep Learning Book*. 2021. Disponível em: <https://www.deeplearningbook.com.br/introducao-as-redes-neurais-convolucionais/>. Acesso em: 26 jun. 2021. Citado na página 22.

DAVIES, A. C.; VELASTIN, S. A. Progress in computational intelligence to support cctv surveillance systems. *International Journal of Computing*, v. 4, n. 3, p. 76–84, 2014. Citado na página 14.

DAVIES, E. R. *Computer and machine vision: theory, algorithms, practicalities*. [S.l.]: Academic Press, 2012. Citado na página 18.

DE, B. Api testing strategy. In: *API Management*. [S.l.]: Springer, 2017. p. 153–164. Citado 2 vezes nas páginas 31 e 32.

DEE, H. M.; VELASTIN, S. A. How close are we to solving the problem of automated visual surveillance? *Machine Vision and Applications*, Springer, v. 19, n. 5, p. 329–343, 2008. Citado na página 14.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2009. p. 248–255. Citado na página 33.

DIEGO, I. et al. Scalable and flexible wireless distributed architecture for intelligent video surveillance systems. *Multimedia Tools and Applications*, v. 78, p. 1–23, 07 2019. Citado 3 vezes nas páginas 35, 37 e 38.

DOLLáR, P. et al. Integral channel features. In: . [S.l.: s.n.], 2009. Citado na página 20.

DOMINGUES, I. et al. Computer vision in esophageal cancer: A literature review. *IEEE Access*, v. 7, p. 103080–103094, 2019. Citado na página 19.

EL TIEMPO. *No para la inseguridad en los buses del SITP, los usuarios expresan sus quejas*. 2021. Disponível em: <https://www.youtube.com/watch?v=3wPIQEqmxuA>. Acesso em: 17 nov. 2021. Citado na página 13.

EVERINGHAM, M. et al. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, v. 88, n. 2, p. 303–338, jun. 2010. Citado na página 41.

G1. *Número de roubos no transporte público do Rio teve aumento de mais de 100% em abril*. 2021. Disponível em: <https://g1.globo.com/rj/rio-de-janeiro/noticia/2021/06/08/numero-de-roubos-no-transporte-publico-do-rio-dobrou-em-abril.ghtml>. Acesso em: 18 nov. 2021. Citado na página 13.

GALAB, M. M.; TAHA, A.; ZAYED, H. Automatic gun detection approach for video surveillance. *International Journal of Sociotechnology and Knowledge Development*, v. 12, p. 49–66, 01 2020. Citado 3 vezes nas páginas 34, 37 e 38.

GIRSHICK, R. Fast r-cnn. 04 2015. Citado na página 24.

GIRSHICK, R. et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. Citado 2 vezes nas páginas 23 e 24.

GONZáLEZ, J. L. et al. Real-time gun detection in cctv: An open problem. *Neural networks : the official journal of the International Neural Network Society*, v. 132, p. 297–308, 09 2020. Citado 3 vezes nas páginas 33, 37 e 38.

HALILI, E. *Apache JMeter*. [S.l.]: Packt Publishing, 2008. ISBN 1847192955. Citado na página 44.

HE, K. et al. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 37, n. 9, p. 1904–1916, 2015. Citado na página 24.

HERRERA, J. et al. Distributed edge cloud r-cnn for real time object detection. In: *2018 World Automation Congress (WAC)*. [S.l.: s.n.], 2018. p. 1–5. Citado 4 vezes nas páginas 15, 35, 37 e 38.

JACCARD, P. Etude de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, v. 37, p. 547–579, 01 1901. Citado na página 28.

KIM, P. Convolutional neural network. In: *MATLAB deep learning*. [S.l.]: Springer, 2017. p. 121–147. Citado na página 23.

KOECH, K. *Object Detection Metrics With Worked Example*. 2020. Disponível em: <https://towardsdatascience.com/on-object-detection-metrics-with-worked-example-216f173ed31e>. Acesso em: 30 jun. 2021. Citado 3 vezes nas páginas 28, 29 e 30.

KUMAR, A. *Gun Detection Dataset with yolo v2 and v3 labels*. 2020. Disponível em: <https://www.kaggle.com/atulyakumar98/gundetection>. Acesso em: 11 set. 2020. Citado 3 vezes nas páginas 20, 39 e 41.

KUZNETSOVA, A. et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020. Citado 3 vezes nas páginas 39, 40 e 41.

LAI, J.; MAPLES, S. *Developing a Real-Time Gun Detection Classifier, World Academy of Science*. [S.l.]: Stanford University, 2017. Citado na página 14.

LEE, S.; KWAK, S.; CHO, M. Universal bounding box regression and its applications. In: SPRINGER. *Asian Conference on Computer Vision*. [S.l.], 2018. p. 373–387. Citado na página 24.

LEE, S. et al. Car plate recognition based on cnn using embedded system with gpu. In: IEEE. *2017 10th International Conference on Human System Interactions (HSI)*. [S.l.], 2017. p. 239–241. Citado na página 18.

LIM, J. et al. Gun detection in surveillance videos using deep neural networks. In: IEEE. *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. [S.l.], 2019. p. 1998–2002. Citado na página 15.

LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: SPRINGER. *European conference on computer vision*. [S.l.], 2014. p. 740–755. Citado na página 43.

LIU, G. et al. Smart traffic monitoring system using computer vision and edge computing. *IEEE Transactions on Intelligent Transportation Systems*, p. 1–12, 2021. Citado na página 19.

LIU, W. et al. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, Springer International Publishing, p. 21–37, 2016. ISSN 1611-3349. Disponível em: <http://dx.doi.org/10.1007/978-3-319-46448-0_2>. Citado 2 vezes nas páginas 15 e 25.

LOHIA, A. et al. Bibliometric analysis of one-stage and two-stage object detection. *Library Philosophy and Practice*, Library Philosophy and Practice, p. 1–32, 2021. Citado na página 23.

MALEKZADEH, T. et al. Aircraft fuselage defect detection using deep neural networks. *arXiv preprint arXiv:1712.09213*, 2017. Citado na página 18.

MILOJICIC, D. S. et al. *Peer-to-peer computing*. [S.l.]: Technical Report HPL-2002-57, HP Labs, 2002. Citado na página 27.

MITHE, R.; INDALKAR, S.; DIVEKAR, N. Optical character recognition. *International journal of recent technology and engineering (IJRTE)*, Citeseer, v. 2, n. 1, p. 72–75, 2013. Citado na página 18.

NEUBECK, A.; GOOL, L. V. Efficient non-maximum suppression. In: *18th International Conference on Pattern Recognition (ICPR'06)*. [S.l.: s.n.], 2006. v. 3, p. 850–855. Citado na página 24.

NEWTON, A. Crime on public transport. In: ____. [S.l.: s.n.], 2014. p. 709–720. ISBN 978-1-4614-5689-6. Citado na página 14.

NIVETHA, S. et al. Recognition and digitization of handwritten text using histogram of gradients and artificial neural network. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, v. 12, n. 6, p. 2555–2564, 2021. Citado na página 18.

NOOR, W. E. I. B. W.; ISA, N. M. et al. Object detection: Harmful weapons detection using yolov4. In: IEEE. *2021 IEEE Symposium on Wireless Technology & Applications (ISWTA)*. [S.l.], 2021. p. 63–70. Citado 3 vezes nas páginas 34, 37 e 38.

OLMOS, R.; TABIK, S.; HERRERA, F. Automatic handgun detection alarm in videos using deep learning. *Neurocomputing*, Elsevier, v. 275, p. 66–72, 2018. Citado 5 vezes nas páginas 15, 33, 34, 37 e 38.

PADILLA, R.; NETTO, S. L.; SILVA, E. A. da. A survey on performance metrics for object-detection algorithms. In: IEEE. *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. [S.l.], 2020. p. 237–242. Citado 4 vezes nas páginas 27, 28, 29 e 30.

PÉREZ-HERNÁNDEZ, F. et al. Object detection binary classifiers methodology based on deep learning to identify small objects handled similarly: Application in video surveillance. *Knowledge-Based Systems*, Elsevier, v. 194, p. 105590, 2020. Citado na página 55.

PETERS, J. *Foundations of Computer Vision*. [S.l.: s.n.], 2017. v. 124. ISBN ISBN 978-3-319-30260-7, ISBN 978-3-319-30262-1 (eBook). Citado 2 vezes nas páginas 18 e 19.

QUEZADA, J. P. A. Robos en transportes públicos colectivos. amenaza a la seguridad pública en méxico. 2019. Citado na página 13.

REDMON, J. *Darknet: Open Source Neural Networks in C*. 2013–2016. <http://pjreddie.com/darknet/>. Citado 2 vezes nas páginas 43 e 44.

Redmon, J. et al. You only look once: Unified, real-time object detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. p. 779–788. Citado 4 vezes nas páginas 25, 34, 42 e 43.

REDMON, J.; FARHADI, A. *YOLO9000: Better, Faster, Stronger*. 2016. Citado na página 25.

REDMON, J.; FARHADI, A. *YOLOv3: An Incremental Improvement*. 2018. Citado na página 25.

REN, J. et al. Distributed and efficient object detection in edge computing: Challenges and solutions. *IEEE Network*, IEEE, v. 32, n. 6, p. 137–143, 2018. Citado na página 15.

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. Citado na página 24.

ROHIT, M. H. An iot based system for public transport surveillance using real-time data analysis and computer vision. In: *2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC)*. [S.l.: s.n.], 2020. p. 1–6. Citado 5 vezes nas páginas 15, 35, 36, 37 e 38.

RUIZ-SANTAQUITERIA, J. et al. *Handgun detection using combined human pose and weapon appearance*. 2020. Citado 3 vezes nas páginas 34, 37 e 38.

SALEHI, N.; KEYVANARA, M.; MONADJEMMI, S. A. An automatic video-based drowning detection system for swimming pools using active contours. *Int. J. Image, Graph. Signal Process*, v. 8, n. 8, p. 1–8, 2016. Citado na página 19.

SANDE, K. et al. Segmentation as selective search for object recognition. In: . [S.l.: s.n.], 2011. p. 1879–1886. Citado na página 24.

SASAKI, Y. The truth of the f-measure. *Teach Tutor Mater*, 01 2007. Citado na página 31.

SHEKHAR, S. *Knife Dataset*. 2020. Disponível em: <https://www.kaggle.com/shank885/knife-dataset>. Acesso em: 11 set. 2020. Citado 3 vezes nas páginas 39, 41 e 55.

SOUZA, T.; CORREIA, S. Estudo de tÉcnicas de realce de imagens digitais e suas aplicaÇÕes. 06 2021. Citado na página 20.

STEEN, M. V.; TANENBAUM, A. S. *Distributed systems*. [S.l.]: Maarten van Steen Leiden, The Netherlands, 2017. Citado 3 vezes nas páginas 25, 26 e 27.

SUN, P. et al. What makes for end-to-end object detection? In: PMLR. *International Conference on Machine Learning*. [S.l.], 2021. p. 9934–9944. Citado na página 24.

SZELISKI, R. *Computer vision: algorithms and applications*. [S.l.]: Springer Science & Business Media, 2010. Citado na página 18.

THUAN. *Gun Detection with YOLO V3*. 2020. Disponível em: <http://doanthuan.com/2020/05/12/gun-detection-with-yolo-v3/>. Acesso em: 11 set. 2020. Citado 2 vezes nas páginas 39 e 40.

TONG, K.; WU, Y.; ZHOU, F. Recent advances in small object detection based on deep learning: A review. *Image and Vision Computing*, v. 97, p. 103910, 2020. ISSN 0262-8856. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0262885620300421>. Citado na página 14.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: SN. *Proceedings of the xxix conference on graphics, patterns and images*. [S.l.], 2016. v. 1, n. 4. Citado na página 23.

VIOLA, P.; JONES, M. J. Robust real-time face detection. *International journal of computer vision*, Springer, v. 57, n. 2, p. 137–154, 2004. Citado na página 21.

VITTORIO, A. *Toolkit to download and visualize single or multiple classes from the huge Open Images v4 dataset*. [S.l.]: Github, 2018. <https://github.com/EscVM/OIDv4_ToolKit>. Citado 2 vezes nas páginas 39 e 41.

WANG, A.; NARAYANAN, A.; RUSSAKOVSKY, O. Revise: A tool for measuring and mitigating bias in visual datasets. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2020. p. 733–751. Citado na página 55.

WANG, K.; BABENKO, B.; BELONGIE, S. End-to-end scene text recognition. In: *2011 International Conference on Computer Vision*. [S.l.: s.n.], 2011. p. 1457–1464. Citado na página 21.

WANG, S.-C. Artificial neural network. In: *Interdisciplinary computing in java programming*. [S.l.]: Springer, 2003. p. 81–100. Citado na página 21.

WARSI, A. et al. Gun detection system using yolov3. In: . [S.l.: s.n.], 2019. p. 1–4. Citado 4 vezes nas páginas 15, 33, 37 e 38.

YAMAZAKI, F. et al. Construction of 3d models of buildings damaged by earthquakes using uav aerial images. In: *Proceedings of the tenth pacific conference earthquake engineering building an earthquake-resilient pacific*. [S.l.: s.n.], 2015. v. 204. Citado na página 18.

ZOU, Z. et al. *Object Detection in 20 Years: A Survey*. 2019. Citado 5 vezes nas páginas 14, 20, 21, 24 e 25.

Apêndices

APÊNDICE A – Exemplo de anotações utilizado no conjunto de dados.

0 0.5264750378214826 0.3709677419354839 0.03177004538577912 0.06048387096774194
0 0.6414523449319214 0.4677419354838709 0.045385779122541 6 0.07258064516129033

APÊNDICE B – Arquivo de classes utilizado para treinamento.

Gun Knife