UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

THIAGO ESTEVES LA SCALA

SMART CAMPUS: UM ESTUDO SOBRE VISÃO COMPUTACIONAL PARA DETECÇÃO DE PESSOAS EM FILAS

BAURU
Janeiro/2023

THIAGO ESTEVES LA SCALA

SMART CAMPUS: UM ESTUDO SOBRE VISÃO COMPUTACIONAL PARA DETECÇÃO DE PESSOAS EM FILAS

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Me. Luiz Felipe de Camargo

BAURU
Janeiro/2023

S279s

Scala, Thiago Esteves La

Smart Campus: um estudo sobre visão computacional para detecção de pessoas em filas / Thiago Esteves La Scala. -- Bauru, 2023

43 p. : il., tabs., fotos

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências, Bauru

Orientador: Luiz Felipe de Camargo

1. Internet das Coisas. 2. Visão Computacional. 3. Aprendizado de Máquina. 4. Detecção de Pessoas. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Thiago Esteves La Scala

Smart Campus: um estudo sobre visão computacional para detecção de pessoas em filas

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Me. Luiz Felipe de Camargo
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Professora Dra Simone Domingues Prado
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Professor Dr Kelton A Pontara da Costa
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Bauru, 18 de janeiro de 2023.

Dedico esse trabalho aos meus familiares que sempre me apoiaram em tudo, principalmete meus pais e avós.

Agradecimentos

Agradeço aos meus familiares pelo incentivo dado para minha educação e por estarem presentes e me apoiarem em tudo durante minha vida universitária.

Agradeço aos meus amigos por sempre estarem presentes nos momentos difíceis e por proporcionarem momentos inesqueciveis na minha vida.

Agradeço também aos professores por tudo que foi ensinado e por sempre estarem dispostos a ajudar em momentos de duvida, em especial meu orientador Prof. Me. Luiz Felipe de Camargo pela disponibilidade e parceria durante a execução do trabalho.

Nossa dor vem da distância entre aquilo que somos e o que idealizamos ser.
Friedrich Nietzsche

Resumo

Considerando as previsões futuras para a área de Internet das Coisas e as facilidades geradas por aplicações de visão computacional utilizando aprendizado de máquina, este trabalho visa a detecção de pessoas em imagens usando o modelo de aprendizado de máquina para detecção de objetos YOLO. Para os testes foram utilizadas modificações do conjunto de dados COCO, composto por diversos tipos de imagens rotuladas, e foram obtidos resultados promissores demonstrados através das métricas índice mAP que alcançou 98,8% e pontuação F1 que alcançou 97%. Os testes desenvolvidos permitiram a análise da aplicabilidade do modelo para a detecção de pessoas, visando o futuro desenvolvimento de uma aplicação para acompanhamento de filas em um Smart Campus.

Palavras-chave: Internet das Coisas, Visão Computacional, Aprendizado de Máquina, Detecção de Pessoas.

Abstract

Considering the future of the Internet of Things area and the ease generated by computer vision applications using machine learning, this work aims to detect people in images using the YOLO object detection model. For the tests, modifications of the COCO dataset were used, composed of several types of labeled images, and promising results were obtained demonstrated through the metrics mAP index that reached 98.8% and F1 score that reached 97%. The developed tests allowed the analysis of the model's applicability for detecting people, aiming at the future development of an application for tracking queues in a Smart Campus.
Keywords: Internet of Things, Computer Vision, Machine Learning, People Detection.

Lista de figuras

Figura 1 – Aparelhos IoT conectados à internet. 17
Figura 2 – Camada de Agrupamento Máximo. 18
Figura 3 – Camada Totalmente Conectada. 19
Figura 4 – Dados dos modelos baseados em *checkpoints* pré treinados. 25
Figura 5 – Gráfico Precisão-Recall do modelo YOLOv5x treinado com o COCO128p. . 28
Figura 6 – Gráfico Pontuação F1 do modelo YOLOv5s com o *dataset* COCO128. . . . 30
Figura 7 – Gráfico Pontuação F1 do modelo YOLOv5x com o *dataset* COCO128. . . . 30
Figura 8 – Gráfico Pontuação F1 do modelo YOLOv5s com o *dataset* COCO128p. . . 31
Figura 9 – Gráfico Pontuação F1 do modelo YOLOv5x com o *dataset* COCO128p. . . 31
Figura 10 – Gráfico Pontuação F1 do modelo YOLOv5s com o *dataset* COCO500. . . . 32
Figura 11 – Gráfico Pontuação F1 do modelo YOLOv5x com o *dataset* COCO500. . . . 32
Figura 12 – Exemplo de detecção do modelo YOLOv5x treinado com COCO128p . . . 34
Figura 13 – Exemplo de detecção do modelo YOLOv5x treinado com COCO500 35
Figura 14 – Exemplo de detecção do modelo YOLOv5x treinado com COCO128p . . . 36
Figura 15 – Exemplo de detecção do modelo YOLOv5x treinado com COCO500 36
Figura 16 – Exemplo de detecção do modelo YOLOv5x treinado com COCO500 37

Lista de tabelas

Tabela 1 – Número de instancias do *label* pessoas em cada *dataset* 24
Tabela 2 – Tempo de treinamneto de cada modelo YOLO 26
Tabela 3 – Código do contador de pessoas na imagem 26
Tabela 4 – Matriz de confusão. 27
Tabela 5 – Indice mAP 29

Lista de abreviaturas e siglas

CNN	Convolutional Neural Network
YOLO	You only look once
IoT	Internet of Things
COCO	Common Objects in Context
mAP	Mean Average Precision
ML	Machine Learning
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
VC	Visão Computacional
AS	Aprendizados Supervisionado

Sumário

1 INTRODUÇÃO 14
1.1 Problemática 14
1.2 Justificativa 14
1.3 Objetivos 15
1.3.1 Objetivo Geral 15
1.3.2 Objetivos Específicos 15
1.4 Organização da Monografia 15

2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 Internet das Coisas 16
2.2 Visão Computacional 17
2.3 Camadas comuns de uma CNN 18
2.3.1 Camada de Convolução 18
2.3.2 Camada de Agrupamento 18
2.3.3 Camada Totalmente Conectada 18
2.4 Aprendizado Supervisionado 19

3 TRABALHOS CORRELATOS 20

4 METODOLOGIA 21
4.1 Materiais 21
4.1.1 YOLO (*You only look once*) 21
4.1.2 COCO (*Common Objects in Context*) 22
4.1.3 Python 22
4.1.4 Pytorch 22
4.1.5 Google Colab 22
4.1.6 Fiftyone 23
4.2 Métodos 23

5 EXPERIMENTAÇÃO E ANÁLISE DOS RESULTADOS 24
5.1 Experimentos 24
5.1.1 Tratamento dos dados 24
5.1.2 Escolha dos modelos 25
5.1.3 Treinamento dos modelos 25
5.1.4 Métricas 27
5.2 Análise dos Resultados 28

6 CONSIDERAÇÕES FINAIS 38
6.1 Trabalhos Futuros 38

REFERÊNCIAS 40

1 Introdução

1.1 Problemática

A comunicação entre pessoas nunca foi tão simples, o número de aparelhos conectados a internet não para de crescer e a velocidade da comunicação está cada vez mais rápida. A internet das coisas, do inglês *Internet of Things* (*IoT*) é, segundo Kumar, Tiwari e Zymbler (2019), o paradigma que permite a comunicação entre aparelhos eletrônicos e sensores através da internet para facilitar as nossas vidas. A *IoT* usa essa comunicação para gerar soluções inteligentes para problemas públicos e privados em todo mundo (SFAR; CHTOUROU; CHALLAL, 2017).

Segundo o relatório da Cisco (2020), quase dois terços da população global terá acesso à internet em 2023 e existirão 3.6 aparelhos conectados à internet *per capta* no mesmo ano.

Existem diversas aplicações de *IoT* sendo elas na indústria, como monitoramento de temperatura para maquinário, na agroindústria, como sistema de irrigação automática, em cidade inteligentes, do inglês *Smart City*, como controle de semáforos e até mesmo em *Smart Campus*. O conceito de *Smart Campus* se refere a aplicações de *IoT* dentro do ambiente universitário. Um exemplo é o *Smart Campus* da Unicamp (UNICAMP, 2016), um projeto que tenta tornar o dia a dia no campus da Unicamp mais produtivo a partir do conceito e aplicações de *IoT*.

1.2 Justificativa

Atualmente o campus da Unesp de Bauru não possui muitas aplicações de *IoT*, um dos poucos exemplos que pode ser citado é o sistema de devolução automático de livros na biblioteca. Tendo em vista isso, a proposta deste projeto é o desenvolvimento de um sistema de reconhecimento de imagem que possa ser usado para diversas aplicações *smart*.

Utilizando visão computacional, é possivel criar um sistema capaz de identificar diversos objetos de acordo com seu treinamento. Esse sistema pode ser usado para inúmeras aplicações de *IoT*, como reconhecimento de placas de carro, monitoramento de vagas de garagem ou monitoramento de filas.

Cada vez mais o uso da tecnologia, com técnicas como o aprendizado de máquina e a visão computacional, vem fazendo parte da vida das pessoas, o auxílio gerado por essas tecnologias tornam as rotinas mais dinâmicas, permitindo um uso mais otimizado do tempo, além de gerar melhorias nos mais diversos processos de gerenciamento, gerando economia. Essas vantagens podem ser obtidas através de uma ferramenta como a proposta nesse trabalho, tornando o processo de esperar em uma fila mais dinâmico e fluído, beneficiando tanto aquele

que espera na fila, como o que deve atender as necessidades daquele que espera.

1.3 Objetivos

1.3.1 Objetivo Geral

Realizar um estudo sobre uma ferramenta de detecção de pessoas em imagens visando a aplicação dela para monitoramento de filas em um *Smart Campus*.

1.3.2 Objetivos Específicos

- Estudar sobre a detecção de objetos com *machine learning*.

- Desenvolver conhecimentos sobre a utilização da biblioteca para aprendizado de máquina Pytorch.

- Desenvolver conhecimentos sobre a utilização do modelo para aprendizado de máquina YOLOv5.

- Treinar um modelo capaz de detectar uma pessoa em uma imagem.

- Através do modelo treinado, contar o número de pessoas presentes na imagem, possibilitando o monitoramento de pessoas em filas.

1.4 Organização da Monografia

Nessa sessão são discutidos os seis capítulos que compõe o trabalho. No Capítulo 1 é apresentada a problemática, justificativa e os objetivos do trabalho. O Capítulo 2 traz toda a fundamentação teórica que foi usada como base para realização do trabalho. No Capítulo 3 são apresentados alguns trabalhos que usam ferramentas semelhantes e tratam dos mesmos temas deste trabalho. O Capítulo 4 discute os materiais e metodos usados para o desenvolvimento e o Capítulo 5 expõe os resultados obtidos e a análise dos mesmos. Por fim, o Capítulo 6 traz as considerações finais e propõe trabalhos que podem ser feitos futuramente.

2 Fundamentação Teórica

Neste capítulo serão apresentados os conceitos teóricos que serviram como base para este trabalho.

2.1 Internet das Coisas

A Internet das Coisas, do inglês *Internet of Things* (*IoT*), é uma rede de objetos físicos, veículos, edifícios e outros itens, equipados com tecnologias de sensor, processamento de dados e conectividade de rede, que permite a eles se conectarem e trocarem dados (FAROOQ et al., 2015). Isso permite que esses objetos "falem" entre si e com as pessoas de forma mais inteligente e automatizada.

A *IoT* pode ser usada em uma ampla variedade de aplicações, como a automação residencial, o monitoramento de saúde, a gestão de transporte e a agricultura de precisão. Por exemplo, uma casa inteligente equipada com sensores de movimento, câmeras de vigilância e controle de temperatura pode ajudar às pessoas a economizar energia, aumentar a segurança e tornar a vida mais conveniente. Em um hospital, os dispositivos IoT, como monitores de sinais vitais e sensores de localização, podem ajudar a gerenciar a assistência médica e a prevenir erros médicos.

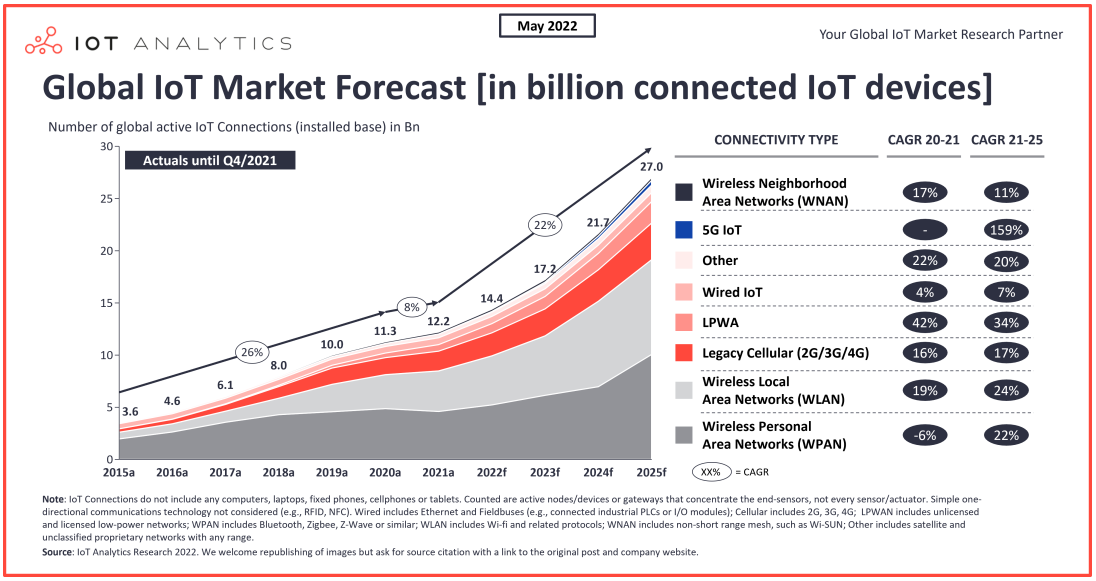
Em resumo, a *IoT* permite que as pessoas interajam de forma mais eficiente com o mundo físico e aumenta a eficiência em diversas tarefas do dia a dia, tornando a vida mais fácil e conveniente.

As aplicações *IoT* são complementadas positivamente pela Visão Computacional (VC). A utilização de cameras em aplicações IoT somada com a implementação de algoritmos de VC permite a identificação de imagens em tempo real, o que pode contribuir para diversos tipos de aplicação.

Um exemplo deste tipo de aplicação é o Sentri360, criada pela empresa Everguard, que é uma espécie de roupa com diversos tipos de sensores, incluindo cameras, que promete diminuir o número de acidentes em ambientes industiais.

> Antes que os trabalhadores andem muito próximo aos caminhões ou carregamentos em andamento, câmeras com visão computacional capturam e coletam os dados, analisam os dados, reconhecem o potencial risco, e dentro de segundos (no máximo), notificam tanto o trabalhador quanto o operador para que parem, por meio de um aparelho vestível (EVERGUARD, 2021).

Figura 1 – Aparelhos IoT conectados à internet.



Fonte: Hasan (2022)

De acordo com Hasan (2022), o mercado de IoT está em contante crescimento e, mesmo tendo desacelerado devido a pandemia de COVID-19, o número de aparelhos IoT conectados deve chegar a 17.2 bilhões em 2023 e até 2025 esse número sobe para 27 bilhões, quase dobrando a previsão de 14.8 bilhões feita para 2022, como visto na Figura 1.

2.2 Visão Computacional

A visão computacional em aprendizado de máquina é o uso da técnica de aprendizado de máquina para permitir que os computadores "vejam" e compreendam o mundo ao seu redor (SHAPIRO; STOCKMAN, 2001).

Isso é alcançado através da análise de imagens e vídeos para extrair informações úteis e tomar decisões baseadas em seu treinamento.

As estruturas mais comuns de aprendizado de máquina para visão computacional incluem redes neurais profundas, do inglês *Deep Learning* (DP), como a rede neural convolucional (CNN), que é especialmente eficaz para reconhecimento de imagem.

A VC pode ser usada em uma ampla variedade de aplicações, como a identificação de objetos e pessoas em imagens e vídeos, a detecção de anomalias ou problemas em equipamentos industriais e a navegação autônoma de veículos.

2.3 Camadas comuns de uma CNN

O YOLO, (*You Only Look Once*) é um modelo de ML considerado estado da arte para detecção de objetos em tempo real (REDMON et al., 2016). Ele é baseado em uma CNN e possui, em sua versão mais básica (YOLOv1), 3 tipos de camadas diferentes, sendo 24 camadas de convolução, 4 camadas de agrupamento máximo e uma *fully conected layer*. Estas camadas sera explicadas nas subseções 2.3.1, 2.3.2 e 2.3.3.

2.3.1 Camada de Convolução

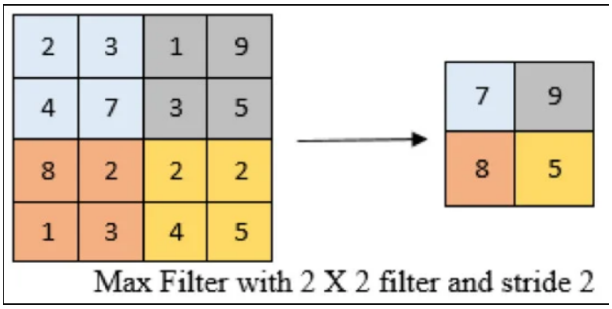
As Camadas de Convolução são as camadas mais importantes de uma CNN. São elas as responsaveis por receber a imagem e usa filtros matriciais (n X n) para destacar informações desejadas nas imagens[1].

2.3.2 Camada de Agrupamento

Após a obtenção dos resultados das camadas de convolução, é necessário reduzir o tamanho espacial do recurso convoluto, a camada de agrupamento é a responsável por essa tarefa. O poder computacional necessário para processar esses dados é reduzida após o agrupamento realizado.

O agrupamento pode ser feito de duas maneiras, mas a usada pelo YOLO é a de *max pooling*. No agrupamento máximo, um kernel de tamanho n X n é movido pela matriz de *pooling* e uma valor máximo é escolhido como saída, como visto na Figura 2.

Figura 2 – Camada de Agrupamento Máximo.



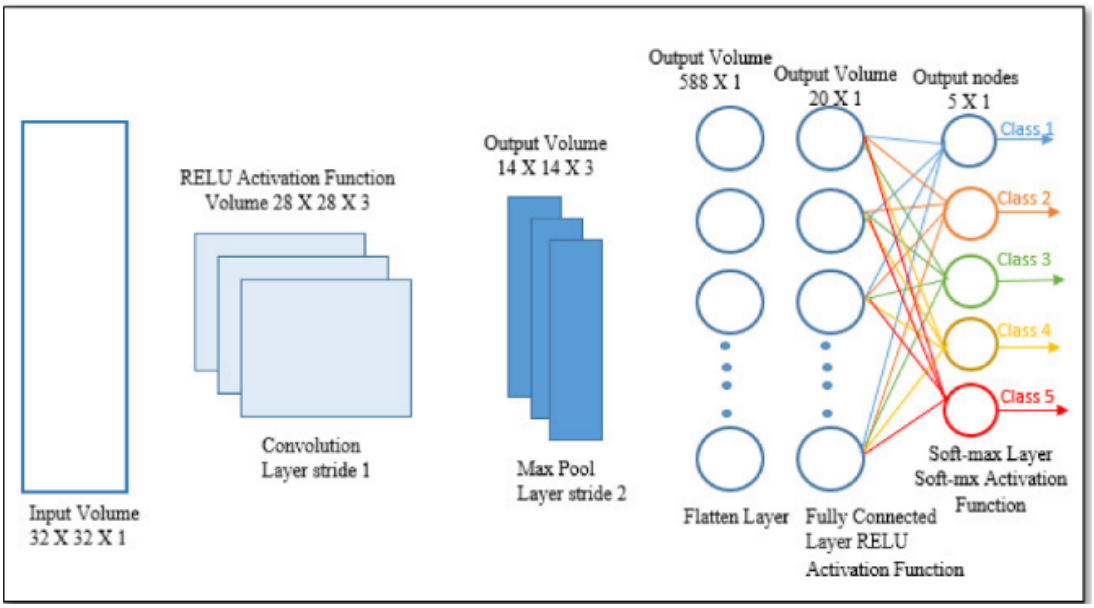
Fonte: Bhatt et al. (2021)

2.3.3 Camada Totalmente Conectada

A camada Totalmente Conectada, ilustrada na Figura 3, é a camada final da rede. Ela recebe as saídas da camada de agrupamento e junta elas para realizar a previsão ou previsões finais da rede.

[1] https://poloclub.github.io/cnn-explainer/

Figura 3 – Camada Totalmente Conectada.



Fonte: Bhatt et al. (2021)

2.4 Aprendizado Supervisionado

Aprendizado de máquina supervisionado (AS) é um ramo de ML para análise de conjuntos de dados rotulados. Algoritmos de aprendizados supervisionado servem para prever resultados a partir dos rótulos anprendidos no treinamento.

Um dos métodos é a AS por classificação, onde o algoritmo é treinado a partir de dados com rótulo de classe, como por exemplo os rótulos "pessoa", "carro", "casa", entre outros.

Além disso, existem diferentes tipos de classificação que um algoritmo pode realizar, como os descritos a seguir (TIBCO, s.d):

- **Classificação Binária** onde os dados de entrada pertencem exclusivamente a dois grupos possíveis.
- **Classificação Multiclasse** onde o treinamento é realizado com muitas classes, porém o dado de saída pertence a somente uma delas.
- **Classificação Multi-rótulo** onde o treinamento é realizado com muitas classes e o dado de saída pode conter múltiplos rótulos.
- **Classificação Desequilibrada** que é semelhante à classificação binária, onde só existem dois tipos possiveis de resultado, porém no treinamento, a quantidade de dados pertencentes a um desses rótulos é muito maior que a quantidade pertencente ao outro.

3 Trabalhos Correlatos

Existem inúmeras aplicações para visão computacional quando se trata de detecção de objetos e neste capítulo seram apresentados alguns trabalhos realizados nessa área.

No trabalho de Baggio, Gonzalez e Borin (2020), foi desenvolvido um sistema que integra as câmeras de um estacionamento na Unicamp com o YOLO. O treinamento foi realizado com imagens tiradas do próprio estacionamento e com isso foi possivel reconhecer e contabilizar o número de vagas disponiveis dentro do estacionamento, facilitando a busca por vagas de carro.

Outras aplicações foram feitas por Lan et al. (2018) e por Laroca et al. (2018), ambos na área de mobilidade urbana usando também o YOLO. O primeiro propõe o uso do YOLO para detecção mais precisa de pedestres. Já o segundo, apresenta o conceito de Reconhecimento Automático de Placas de Veículos (ALPR) e como o YOLO pode ser usado para esse tipo de função, realizando um treinamento com videos de carros, motocicletas, ônibus e caminhões, sendo uma aplicação muito importante para o monitoramento de trânsito.

George et al. (2018) propõem o uso de uma rede de aprendizado profundo baseada em YOLO para detecção de nódulos pulmonares durante tomografias computadorizadas de baixa dosagem em tempo real, os resultados mostraram-se promissores por possuir taxas baixas de falsos positivos com alta precisão, sendo uma contribuição muito importante para a área de saúde.

Ainda na área de saúde, o trabalho de Lemay (2019), que também analisa imagens de tomografias computadorizadas, porém foca em detectar em tempo real a localização dos rins. A detecção de orgãos em tempo real pode ser favorável para radioterapia adaptativa e para cirurgias, como as cirurgias laparoscópicas onde a incisão é minima e pouco invasiva.

Durante o surto global de coronavírus (COVID-19), o distanciamento social virou norma para tentar combater a progressão do virus. Ahmed et al. (2021) criou um sistema de rastreamento de distanciamento social. Esse sistema tem como base um algoritmo de detecção YOLOv3 capaz de detectar pessoas e calcular a distância entre as pessoas detectadas na imagem. Esse cálculo é feito a partir da distância euclidiana dos centros das caixas delimitadoras de duas pessoas. Usando uma aproximação da distância física com a distância de pixels na imagem foi possivel calcular a distância entre as pessoas e decidir se houve ou não uma violação de distanciamento social com uma precisão de 95%.

4 Metodologia

Neste capítulo serão apresentados as principais ferramentas e tecnologias utilizadas para a composição do trabalho assim como os métodos utilizados para a realização do mesmo.

4.1 Materiais

4.1.1 YOLO (*You only look once*)

YOLO (*You only look once*) é um algoritmo de detecção de objetos usado em visão computacional que foi criado para ser rápido e preciso. Ele permite detectar e classificar objetos em imagens e vídeos em tempo real.

O YOLO é estruturado em duas partes principais: uma rede neural treinada para prever as *bounding boxes* (caixas delimitadoras) e classes dos objetos em uma imagem, e um algoritmo de processamento que usa essas previsões para detectar objetos na imagem (REDMON et al., 2016). A rede neural é treinada usando uma grande quantidade de imagens rotuladas manualmente, contendo exemplos de todos os objetos que o YOLO deve ser capaz de detectar.

Uma vez treinada, a rede neural é capaz de processar uma imagem de entrada e gerar uma série de previsões, cada uma contendo uma *bounding box* e uma classe de objeto. O algoritmo de processamento então usa essas previsões para detectar os objetos na imagem, eliminando as caixas delimitadoras que se sobrepõem ou que são muito pequenas ou grandes em relação ao tamanho dos objetos na imagem. Isso permite que o YOLO detecte vários objetos diferentes em uma única imagem de forma rápida e precisa.

Existem várias versões do YOLO, incluindo YOLOv1 (publicado em 2015), YOLOv2 (publicado em 2016), YOLOv3 (publicado em 2018) e YOLOv4 (publicado em 2020). YOLOv5 (ULTRALYTICS, 2020) é a última versão, disponível para o público, do YOLO, publicada em 2020. Ela foi aperfeiçoada com o objetivo de aumentar ainda mais a precisão e velocidade do algoritmo. Além disso, a YOLOv5 inclui várias novas características, como suporte a vários unidades de processamento de vídeo (GPUs) e aceleração por *hardware*, que permitem que ela seja executada de forma ainda mais rápida em dispositivos de alta capacidade de processamento.

YOLOv5s é uma variante da YOLOv5 que foi otimizada para ser mais rápida, enquanto mantém alta precisão. Ela é menor do que a YOLOv5 e usa menos recursos de processamento, tornando-a adequada para ser executada em dispositivos de menor poder de processamento. Esse modelo possui 214 *layers*, 7.235.389 parâmetros, 7.235.389 gradientes e 16.6 GFLOPs.

YOLOv5x é uma variante da YOLOv5 que foi otimizada para ser ainda mais precisa, mesmo quando detectando objetos de tamanhos pequenos ou distorcidos. Ela é mais precisa

do que a YOLOv5s, mas também é mais lenta e usa mais recursos de processamento. Esse modelo possui 445 *layers*, 86.749.405 parâmetros, 86.749.405 gradientes e 206.3 GFLOPs.

4.1.2 COCO (*Common Objects in Context*)

COCO é um conjunto de dados, também chamado de *dataset*, criado pela Microsoft (2015) que possui mais de 200 mil imagens contendo mais de 80 categorias de objetos. O principal objetivo do conjunto de dados é facilitar na detecção de objetos por coordenadas na imagem, através de caixas delimitadoras, e permitir a diferenciação dos objetos presentes em uma determinada imagem.

4.1.3 Python

A linguagem de programação Python foi escolhida por ser *open source*, por ser uma das linguagens mais populares na área de *machine learning* (RASCHKA; PATTERSON; NOLET, 2020) e por ser facilmente usada com o YOLO e com o Google Colab.

Devido a isso, ela possui várias *frameworks* e bibliotecas que auxiliam no trabalho com ML, como o Pytorch e TensorFlow.

4.1.4 Pytorch

Pytorch é uma biblioteca Python *open source* de ML, desenvolvida pelo laboratório de pesquisa em Inteligencia Artificial do Meta, usada para implementação de aprendizado profundo (LEMOS, 2022).

Por possuir capacidades de implementação de *deep learning*, ela é ideal para ser usada em aplicações de visão computacional, mais expecificamente em detecção de objetos.

4.1.5 Google Colab

Google Colaboratory, mais conhecido como Google Colab (GOOGLE, s.d), é um serviço de nuvem gratuito, hospedado pela Google, onde é possivel ter acesso a *notebooks*, como são chamados os ambientes de prototipação utilizando a linguagem Python, de alto poder computacional de forma gratuita ou por assinatura mensal.

O ambiente do Colab trabalha especialmente com a linguagem Python e é muito semelhante ao *Jupyter Notebook*, permitindo segmentação do código na forma de células. A ferramenta facilita muito o tratamento de grandes quantidades de dados por ser hospedada em nuvem e ter acesso à GPU's, independente do *hardware* da sua máquina local.

A utilização de uma GPU acelera o tempo de treinamento quando comparado a uma CPU (STEINKRAUS; BUCK; SIMARD, 2005), permitindo executar algoritmos complexos em qualquer tipo de máquina.

4.1.6 Fiftyone

Fiftyone (VOXEL51, s.d) é uma ferramenta *open source* para visualização e tratamento de dados desenvolvida pela empresa Voxel51 na linguagem Python.

A ferramenta possui integração nativa com diversas outra ferramentas sendo as principais o Google Colab, Pytorch e o *dataset* COCO.

Por permitir o tratamento de dados, é possivel abrir um *dataset* já existente, como o COCO, e modificá-lo para atender as necessidades do usuário. Neste trabalho foram usados três *datasets*, dos quais dois deles são modificações do COCO. O primeiro é um *dataset* que possui as primeiras 128 imagens que possuem o *label "person"* e o segundo, os primeiros 500 casos do mesmo *label*.

4.2 Métodos

Inicialmente foi realizado um levantamento bibliográfico sobre a área de visão computacional e aprendizado de máquina, aprofundando em pesquisas sobre aprendizado profundo e CNN. Um estudo para compreender os diversos modelos e aplicações de visão computacional também foi realizado durante esta fase. Nesta etapa também foi escolhido o modelo a ser usado para o desenvolvimento do trabalho.

Posteriormente, foram escolhidas as ferramentas e o *dataset* que auxiliaram no treinamento do modelo escolhido previamente. As ferramentas de auxílo escolhidas foram o Google Colab e o Fiftyone e o dataset escolhido foi o COCO. A linguagem de programação escolhida foi o Python por ser uma linguagem que abrange muito a área de aprendizado de máquina e por sua simples aplicação com o modelo escolhido, o YOLOv5.

Por fim, foram realizados os treinamentos dos modelos do YOLO com três *datasets* preparados, gerando um conjunto de resultados para análise e comparação.

5 Experimentação e Análise dos Resultados

5.1 Experimentos

5.1.1 Tratamento dos dados

Como a plataforma Google Colab apresenta limitação no tempo de execução, foi necessário adequar o *dataset* para que o treinamento possa ser realizado na versão gratuita do Colab.

A partir disso, foram usados três *datasets* derivados do COCO. O primeiro deles, chamado de COCO128, é um *dataset* que possui as 128 primeiras imagens presentes no COCO original. O segundo é semelhante ao primeiro, possuindo também 128 imagens, porém essas imagens são as primeiras 128 que possuem o *label "person"*, nomeado de COCO128p. Por ultimo, o terceiro possui 500 imagens, sendo elas as primeiras 500 com o mesmo *label*, nomeado de COCO500. A Tabela 1 mostra no número de instancias do rótulo pessoas em cada conjunto de dados.

Tabela 1 – Número de instancias do *label* pessoas em cada *dataset*

Dataset	Labels Pessoa
COCO128	254
COCO128p	537
COCO500	2109

Fonte: Elaborada pelo autor

Os dois últimos *datasets* foram criados, com o auxilo da ferramenta Fiftyone[1], para tentar melhorar os resultados do COCO128. A decisão de filtrar as imagens por *label* foi um teste para ver se a especificidade dos dados faria diferença nos resultados. Posteriormente, criou-se o conjunto de dados com 500 imagens para testar se a quantidade dos dados afetaria os resultados. O número 500 foi escolhido por ser relativamente maior que 128 e por ter encaixado dentro do limite de 5 horas de treinamento.

Inicialmente os treinamentos foram realizados com 100 épocas, porém devido ao baixo número de dados comparado ao inicio, optou-se por aumentar o número de épocas que passou a ser 300. Portanto, todos os treinamentos foram realizados com 300 épocas nos modelos YOLOv5s e YOLOv5x.

[1] https://voxel51.com/docs/fiftyone/user_guide/export_datasets.html#yolov5dataset-export

5.1.2 Escolha dos modelos

O YOLO possui diversos modelos para treinamento sendo os pricipais o YOLOV5n, YOLOv5s, YOLOv5n, YOLOv5l e YOLOv5x. Dentre as suas diferenças pode-se destacar a precisão e a velocidade como principais, tendo o YOLOv5n como o mais veloz e o YOLOv5x como o mais preciso (ULTRALYTICS, 2020).

Figura 4 – Dados dos modelos baseados em *checkpoints* pré treinados.

Model	size (pixels)	mAP^{val} 50-95	mAP^{val} 50	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	**45**	**6.3**	**0.6**	**1.9**	**4.5**
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

Fonte: Ultralytics (2020)

Os modelos escolhidos para o desenvolvimento do trabalho foram o YOLOv5s e o YOLOv5x. O primeiro foi escolhido por sua velocidade, quando comparado com o YOLOv5n, a diferenca de velocidade entre os dois foi considerada semelhante (6.3ms para o YOLOv5n e 6.4ms para o YOLOv5s), enquanto usada uma GPU, sendo a precisão entre os dois o fator decisivo na escolha do YOLOv5s. Já o YOLOv5x foi escolhido exclusivamente por ser o mais preciso, como visto na Figura 4. A diferença de precisão pode ser atribuída ao número de camadas entre os dois ser diferente, como mostrado na subsessão 4.1.1.

Foram escolhidos dois modelos diferentes para que possam ser gerados resultados para ambos e ao final do trabalho esses dados foram comparados e foram discutidas as diferenças entre eles e se há vantagem ter um treinamento mais demorado para obtenção de tais resultados.

5.1.3 Treinamento dos modelos

Inicialmente os treinamentos foram realizados em uma máquina local, CPU @2.80GHz e 8GB RAM, porém isso logo mudou, uma vez que os treinamentos davam erro de memória. Devido a isso, passou-se a usar o Google Colab como plataforma para realização dos treinamentos, por ser possivel realizá-los com uma GPU de *back-end* da Google e com uma memória de 16GB disponível.

O treinamento foi iniciado usando o *dataset* COCO nos modelos YOLOv5s e YOLOv5x com 100 épocas e com um *batch size* de 16, mas o treinamento foi logo interrompido devido

ao tempo previsto para concluir uma única época do YOLOv5s ter sido de 1 hora 27 minutos, o que viria a ser inviável devido a limitação de 12 horas do Colab.

A tentativa frustrada de treinamento, mostrou que seria necessário diminuir significamente a quantidade de dados utilizados devido as limitações encontradas. Como foi discutido na subsessão 5.1.1, outros 3 *datasets* foram utilizados para contornar o problema do tempo.

Após a escolha dos novos conjuntos de dados e depois dos primeiros testes, o número de épocas passou a ser 300 devido a diminuição drástica na quantidade de dados. Portanto, os treinamentos passaram a ser feitos com 300 épocas nos modelos YOLOv5s e YOLOv5x e com os *datasets* COCO128, COCO128p e COCO500, na plataforma Google Colab gratuita.

Tabela 2 – Tempo de treinamneto de cada modelo YOLO

Modelo	Dataset	Tempo de Treinamento
YOLOv5s	COCO128	1 hora e 34 minutos
	COCO128p	1 hora e 35 minutos
	COCO500	3 horas e 11 minutos
YOLOv5x	COCO128	2 horas e 47 minutos
	COCO128p	2 horas e 46 minutos
	COCO500	4 horas e 40 minutos

Fonte: Elaborada pelo autor

A Tabela 2 mostra quanto tempo levou o treinamento de cada *dataset* nos modelos escolhidos com 300 épocas.

Por fim, foi adicionado o código (Tabela 3), que serve para adicionar o contador de pessoas na imagem, no arquivo de código , detect.py, arquivo este responsavel por detectar os objetos nas imagens.

Tabela 3 – Código do contador de pessoas na imagem

```
1  class_name_count = 'person'# variavel para procurar por um label
2  l = s[1:s.find(class_name_count)].split()[-1] # variavel contador do label
3  if class_name_count in s: # se encontrar o label entra no if
4          print(l,class_name_count) #printa o contador e o nome do label no console
5          cv2.rectangle(im0, (0,0), (150,40), (0,0,0), -1)
           #printa um retangulo para contraste na imagem
           cv2.putText(im0,l+ class_name_count,(0,30),
6          cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255),2,cv2.LINE_AA)
           #printa o contador e o nome do label na imagem
```

Fonte: Elaborada pelo autor

5.1.4 Métricas

Nessa subseção são discutidas as métricas usadas para analisar os resultados sendo elas a mAP e a pontuação F1, que, segundo Yohanandan (2020) e Bajaj (2022), são métricas muito usadas para medir a qualidade de modelos de detecção de objetos.

Para definir essas métricas é necessário ter uma matriz de confusão dos resultados. A matriz de confusão é uma matriz que contém a relação entre as classes reais dos objetos e o resultado que foi previsto. Essa relação é dividida entre Positivo Verdadeiro (TP), Positivo Falso (FP), Falso Negativo (FN) e Falso Positivo (TN).

Tabela 4 – Matriz de confusão.

Classe real/Resultado	Positivo	Negativo
Positivo	TP	FN
Negativo	FP	TN

Fonte: Elaborada pelo autor

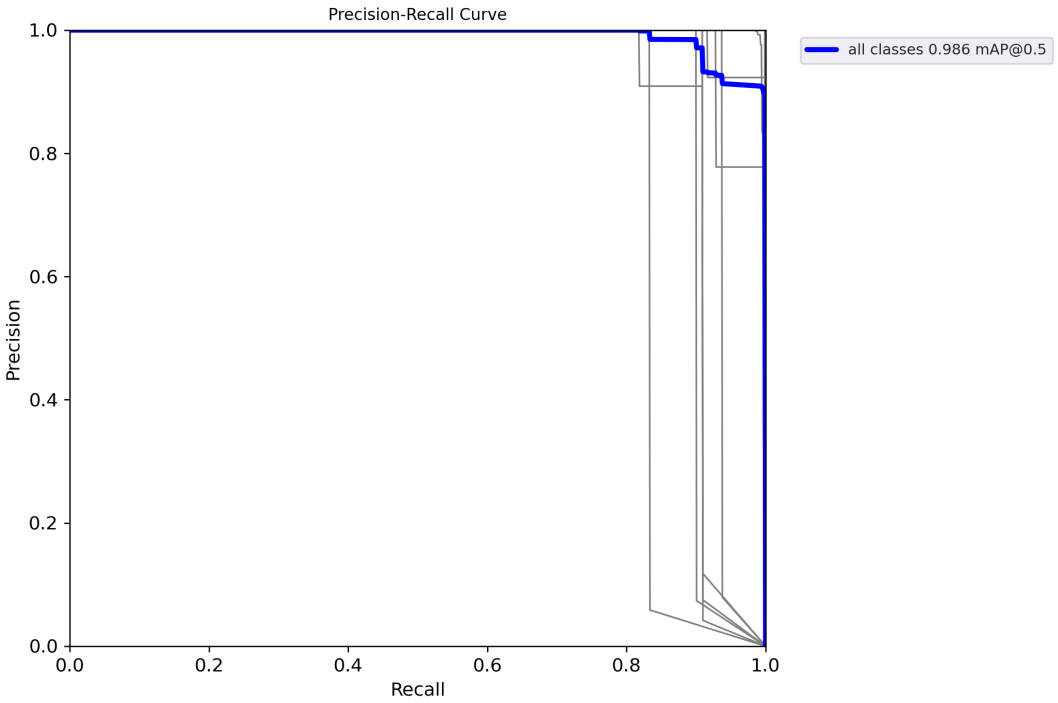
A partir da matriz de confusão (Tabela 4) é possível calcular a precisão (P) dos resultados e o *recall* usando as equações 5.1 e 5.2 respectivamente.

$$P = \frac{TP}{TP + FP} \tag{5.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

Com esses dois valores definidos para cada época pode-se criar um gráfico de Precisão-Recall. Como o gráfico da Figura 5. A partir da interpolação dos pontos desse gráfico pode-se calcular a área sob a curva, adquirindo então a medida do AP (*Average Precision*).

Figura 5 – Gráfico Precisão-Recall do modelo YOLOv5x treinado com o COCO128p.



Fonte: Elaborada pelo autor.

Com os valores de AP, o mAP é calculado com a somatória dos APs dividido pelo número de classes no modelo treinado.

Já a pontuação F1 é calculada a partir da média harmônica entre precisão e o *recall*, como mostrado na equação 5.3.

$$F1 = \frac{2 * P * Recall}{P + Recall} \tag{5.3}$$

5.2 Análise dos Resultados

Os modelos estudados apresentam comportamentos diferentes quando aplicados sobre os 3 conjuntos de dados.

O YOLOv5s aparenta beneficiar-se da utilização de *datasets* pequenos, e apresenta melhores resultados no conjunto com poucos dados e com mais *labels* de pessoas.

O YOLOv5x apresenta melhores resultados no *dataset* maior, embora a diferença dos resultados apresentados seja pequena. Este modelo também realiza melhores previsões quando treinado com mais informações pertinentes à caracterização de pessoas.

O maior modelo apresentou os melhores desempenhos em todos os treinamentos realizados, e o pior resultado que oferece é maior que o melhor do menor modelo.

No entanto, mesmo com diferenças, todos os resultados são positivos para esta métrica (acima de 90% de mAP) e o que mais diferencia os dois modelos é o tempo de treinamento.

É importante analisar se a situação de implementação dos modelos permite e/ou exige o sacrifício de alguns pontos de precisão para poupar tempo de treinamento do sistema. Os resultados podem der vistos na Tabela 5.

Tabela 5 – Indice mAP

Modelo	Dataset	mAP
YOLOv5s	COCO128	0.969
	COCO128p	0.973
	COCO500	0.958
YOLOv5x	COCO128	0.978
	COCO128p	0.986
	COCO500	0.989

Fonte: Elaborada pelo autor

A seguir estão os gráficos de curva F1-Confiança, esses gráficos apresentam uma comparação entre o grau de confiança atribuido na detecção e a pontuação F1, todos possuem basicamente a mesma forma, começando baixo, atingindo um ponto maximo e caindo ao se aproximar de 100% de confiança.

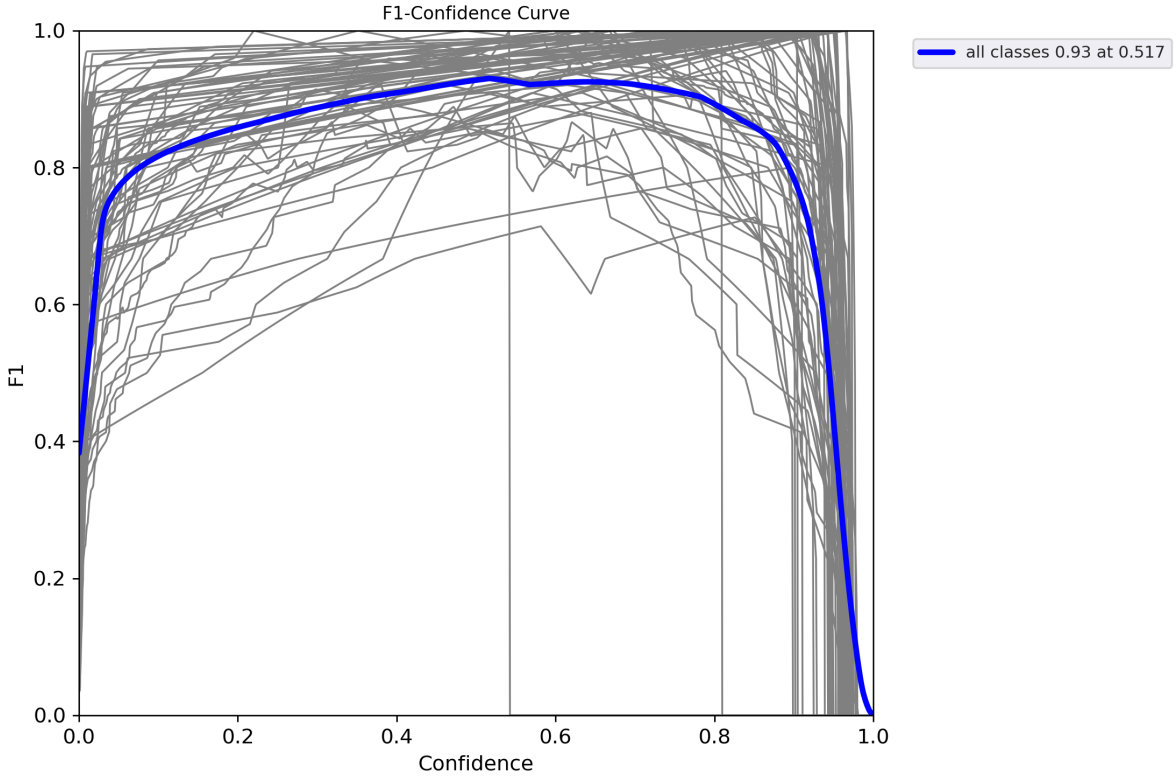
Na Figura 6 tem-se o gráfico do modelo YOLOv5s treinado com o COCO128. Aqui pode-se observar que a pontuação F1 inicia em aumento e chega em seu valor máximo de 93% quando o grau de confiança é de 51.7% caindo logo em seguida.

O modelo YOLOv5x (Figura 7) treinado, também treinado com o COCO128, diferente do anterior, tem a pontuação máxima de F1 (96%) em 75.6% de confiança.

Nas Figuras 8 e 9 estão os resultados do treinamento com o COCO128p dos modelos YOLOv5s e YOLOv5x respectivamente. Ambos possuem um aumento de pontuação F1 quando comparados ao *dataset* anterior, o primeiro com pico em 66.5% de confiança e novamente 93% de pontuação F1 e o segundo em 79.2% com 97%.

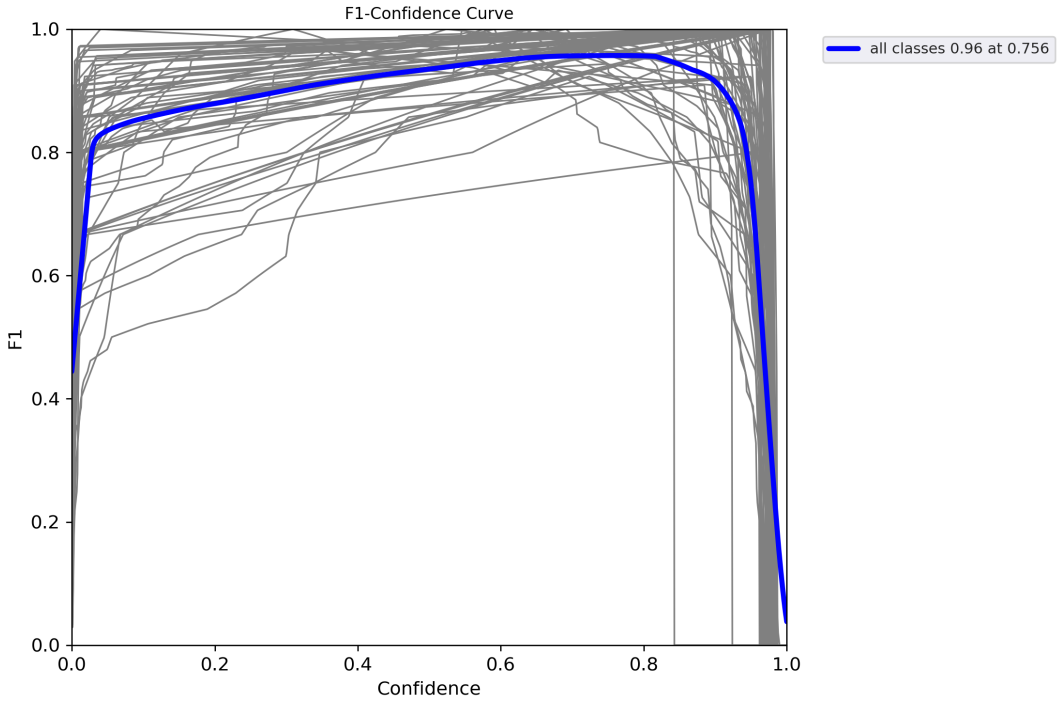
Finalmente, as Figuras 10 (YOLOv5s) e 11 (YOLOv5x) treinados no COCO500 tem resultados identicos aos anteriores de pontuação F1, 93% e 97%, porém tem seus picos em 52% e 65% de confiança.

Figura 6 – Gráfico Pontuação F1 do modelo YOLOv5s com o *dataset* COCO128.



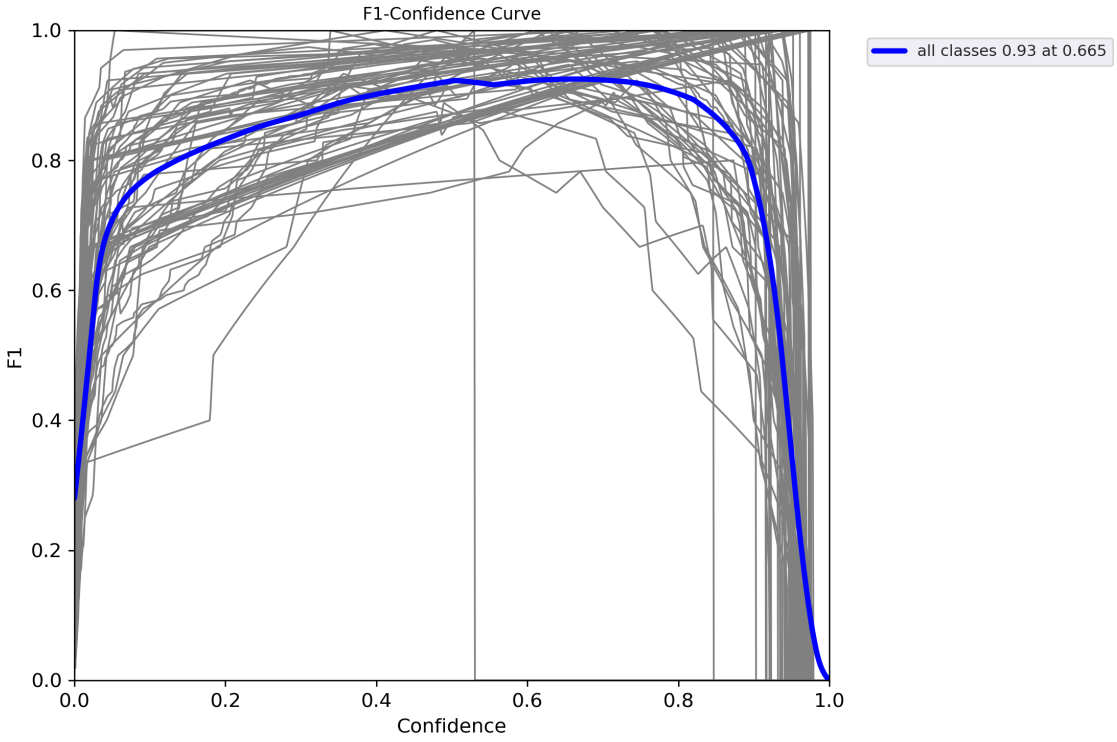
Fonte: Elaborada pelo autor.

Figura 7 – Gráfico Pontuação F1 do modelo YOLOv5x com o *dataset* COCO128.



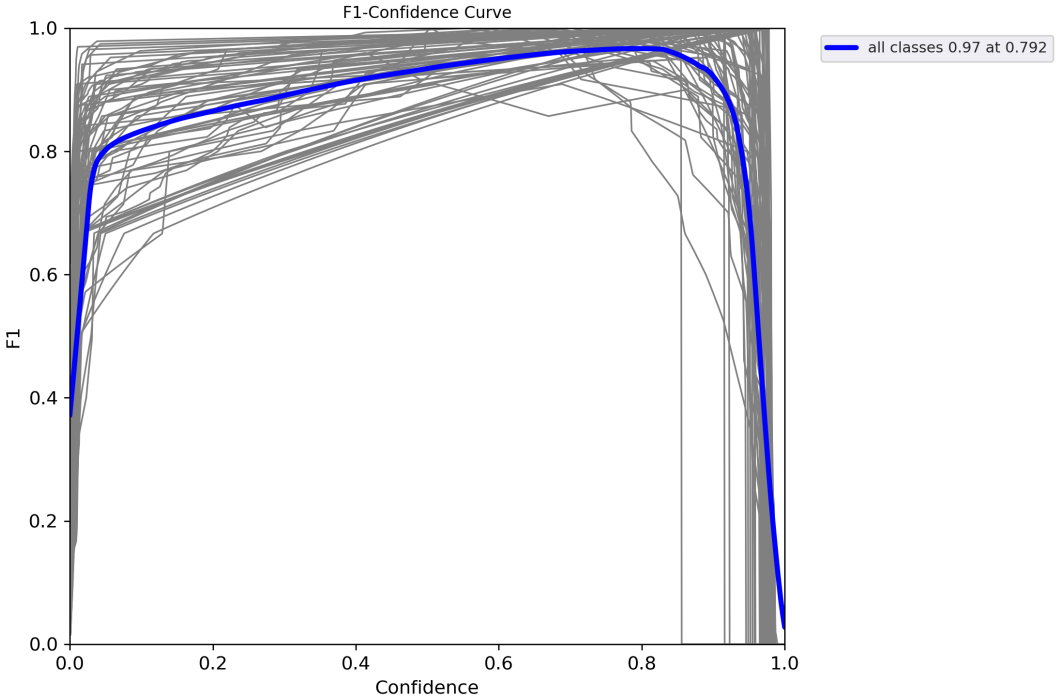
Fonte: Elaborada pelo autor.

Figura 8 – Gráfico Pontuação F1 do modelo YOLOv5s com o *dataset* COCO128p.



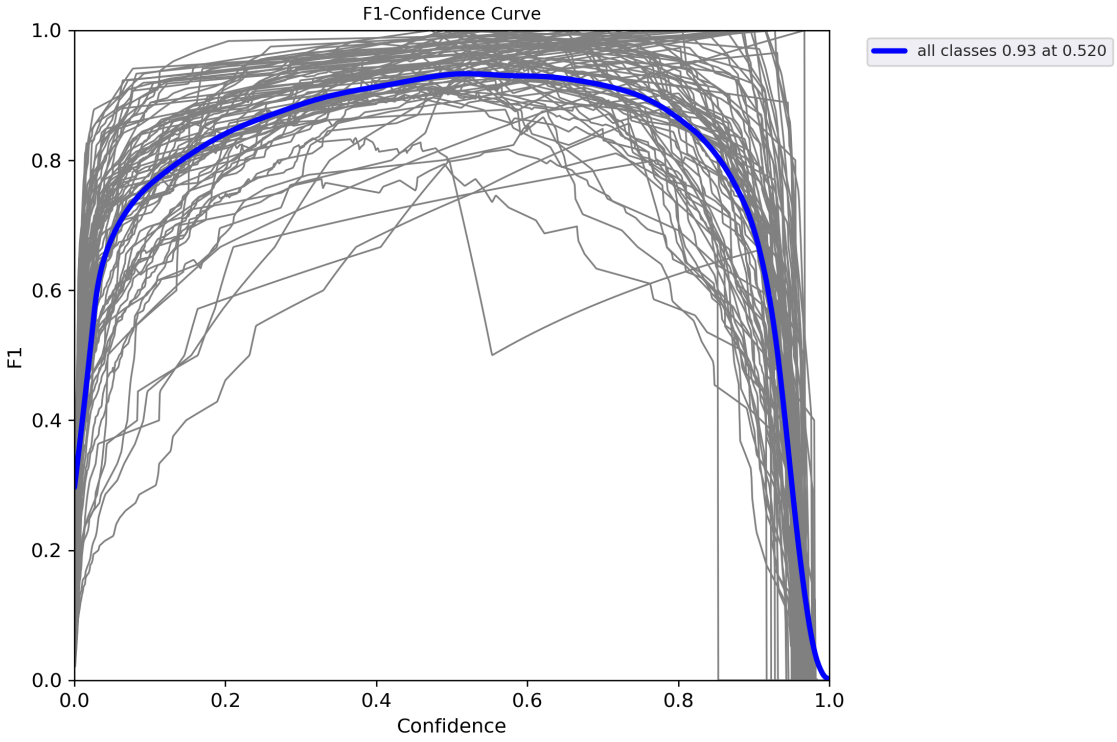
Fonte: Elaborada pelo autor.

Figura 9 – Gráfico Pontuação F1 do modelo YOLOv5x com o *dataset* COCO128p.



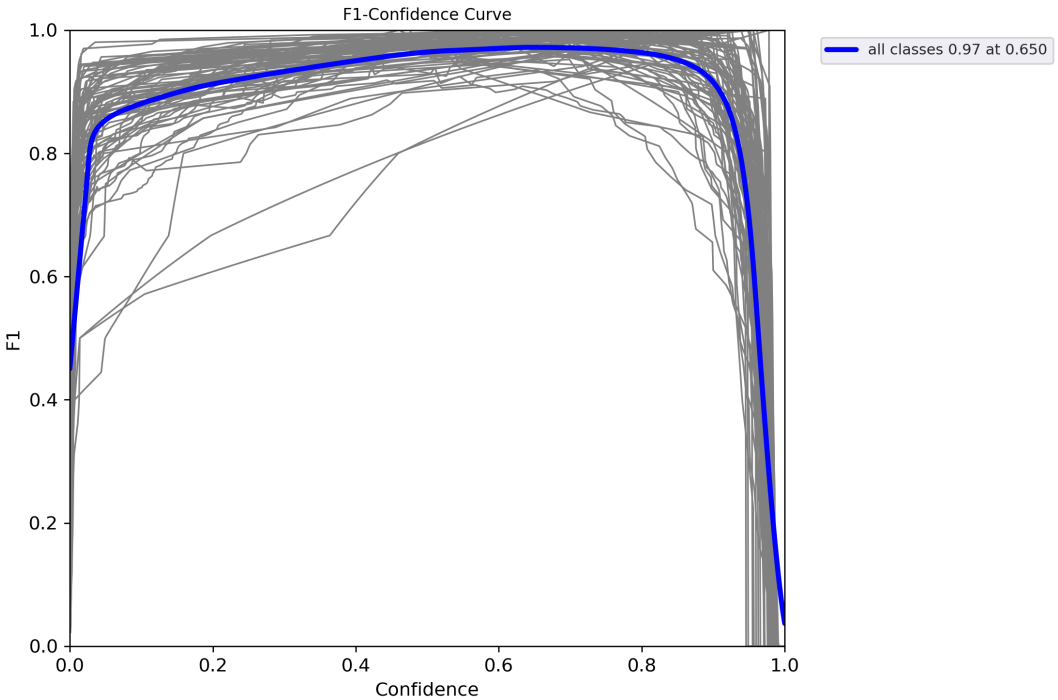
Fonte: Elaborada pelo autor.

Figura 10 – Gráfico Pontuação F1 do modelo YOLOv5s com o *dataset* COCO500.



Fonte: Elaborada pelo autor.

Figura 11 – Gráfico Pontuação F1 do modelo YOLOv5x com o *dataset* COCO500.



Fonte: Elaborada pelo autor.

Considerando os resultados obtidos pelas métricas F1 e mAP, os melhores modelos treinados foram o YOLOv5x com o *dataset* COCO128p e com o COCO500.

Analisando as Figuras 12, 13, 14 e 15, pode-se observar uma oposição. Enquanto nas imagens 12 e 13 o treinamento realizado com o conjunto de dados COCO128p teve mais detecções e mais grau de confiança que o COCO500, nas Figuras 14 e 15 o oposto foi encontrado, com o modelo treinado com o COCO500 detectando mais pessoas com graus de confiança semelhantes ao do COCO128p.

Como foi visto com os resultados de mAP, ambos os treinamentos possuem resultados semelhantes, não possuindo diferenças marcantes de um para outro.

Analisando com a pontuação F1, a Figura 12 tem mais previsões com grau de confiança próximo ao grau responsável pelo valor máximo de F1, em comparação à Figura 13. Já entre as Figuras 14 e 15, ambos possuem pontuações F1 semelhantes.

Como foi discutido anteriormente, é preciso analisar cada situação de forma individual para decidir se é viável utilizar um tempo de treinamento maior para atingir uma precisão maior ou se a precisão não é tão importante quanto a velocidade de treinamento.

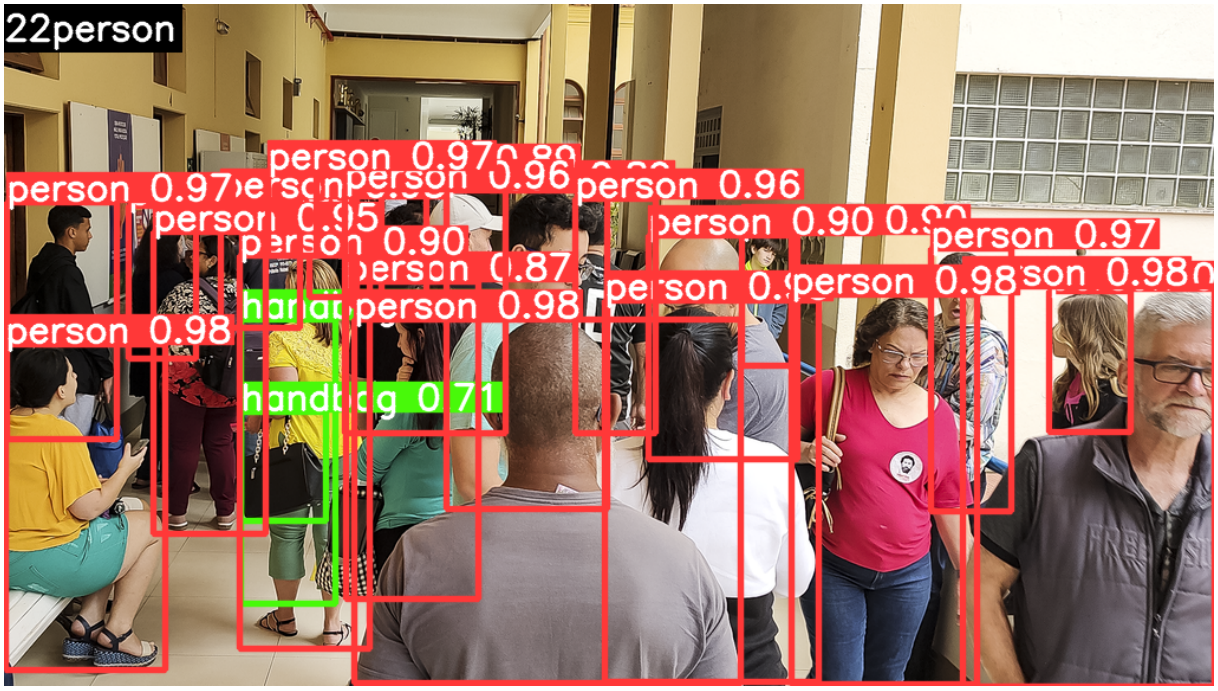
Figura 12 – Exemplo de detecção do modelo YOLOv5x treinado com COCO128p

Fonte: Imagem retirada de Images (2011) e adaptada pelo autor.

Figura 13 – Exemplo de detecção do modelo YOLOv5x treinado com COCO500

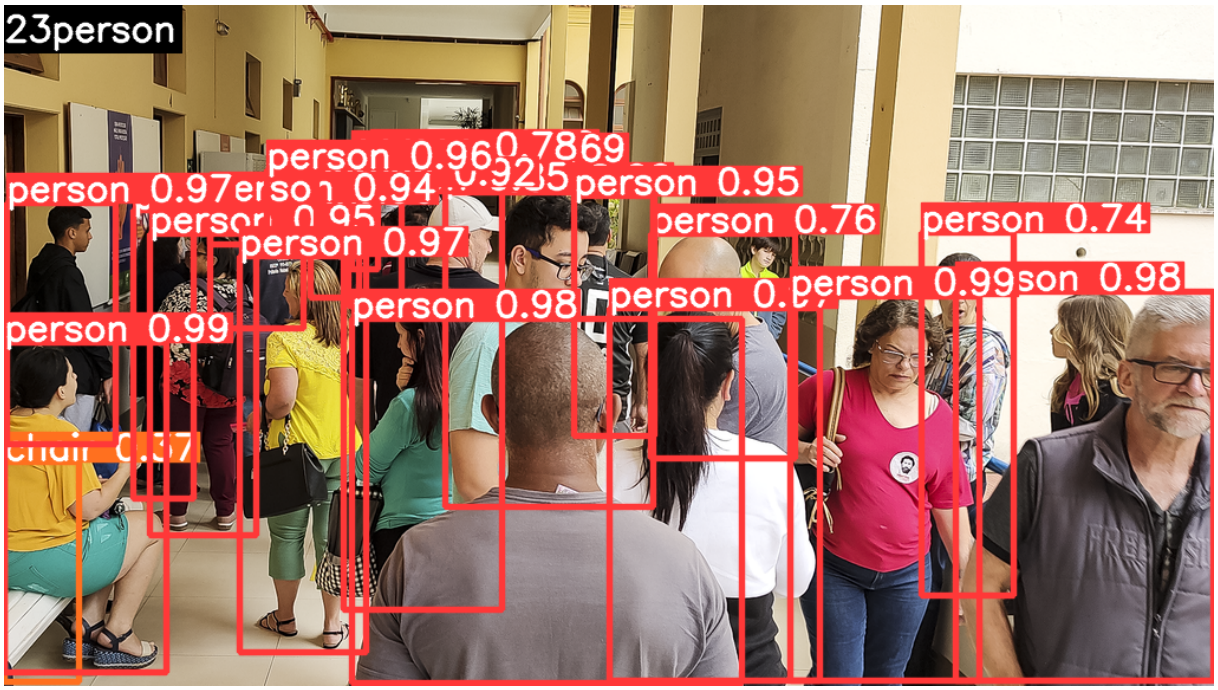
Fonte: Imagem retirada de Images (2011) e adaptada pelo autor.

Figura 14 – Exemplo de detecção do modelo YOLOv5x treinado com COCO128p



Fonte: Imagem retirada de Espíndola, Amorim e Coutinho (2022) e adaptada pelo autor.

Figura 15 – Exemplo de detecção do modelo YOLOv5x treinado com COCO500



Fonte: Imagem retirada de Espíndola, Amorim e Coutinho (2022) e adaptada pelo autor.

Existem limitações perceptíveis nos modelos treinados, pessoas distantes da camera e pessoas parcialmente ocultas por outras pessoas ou por objetos acabam passando, na maioria

das vezes, despercebidas, como é possivel ver na Figura 12 onde o número de detecções de pessoas localizadas do meio para o fim da fila é menor do que quando as pessoas estão mais na frente (perto da camera)

Esse problema pode ser contornado com um melhor posicionamento da camera como visto na Figura 16 ou com um treinamento com dados mais especificos que foquem nos pontos fracos de detecção.

Figura 16 – Exemplo de detecção do modelo YOLOv5x treinado com COCO500

Fonte: Imagem retirada de G1 (2018) e adaptada pelo autor.

6 Considerações Finais

Como visto nesse trabalho o YOLO possui inumeras aplicações e pode ser usado de maneiras quase ilimitadas. Existem muitos trabalhos feitos e o número de informações disponíveis na Internet é bem abrangente.

O algoritmo tem uma boa performance com reconhecimento de pessoas e imagens de pessoas e fotos são abundantes na internet, portanto a obtenção de *datasets* que contenham pessoas é de fácil acesso.

O YOLO é bem intuitivo e de fácil acesso, mesmo sem um *hardware* adequado para rodá-lo é possível obter resultados satisfatórios com treinamento limitado e a existência de plataformas para execução *online* facilita ainda mais, permtindo um treinamento mais complexo sem a disponibilidade de hardware físico.

A integração de sistemas físicos com o reconhecimento de imagem abre portas para diversas aplicações loT que podem ser usadas para a melhoria na qualidade de vida das pessoas.

6.1 Trabalhos Futuros

Várias melhorias e alterações podem ser feitar neste trabalho para melhorar o desempenho e os resultados obtidos, algumas delas que podem ser citadas são:

- Buscar outros *datasets* para realizar o treinamento e comparar os resultados obtidos.
- Pesquisar outros modelos diferentes do YOLO e realizar testes com eles.
- Melhorar o treinamento com os modelos atuais, alterando a quantidade e qualidade dos dados de treinamento.
- Obter uma máquina local com hardware melhor que permita a utilização de mais dados.
- Contratar a versão Pro ou Pro+ da plataforma Google Colab, assim evitando as limitações da versão gratuita.
- Estudar sobre o Kaggle e configurar ele para substituir o Google Colab. Kaggle possui limitação de 30 horas de GPU.
- Integração com uma camera fisica para reconhecimento em tempo real.
- Desenvolver uma interface para aplicação em tempo real.

Se possível, realizar uma parceria com a UNESP para que esse projeto e outros possam ter apoio e financiamento para serem implementados na própria universidade trazendo melhorias diretas para a universidade pública, podendo talvez tornar um projeto semelhante ao Smart Campus - Unicamp (UNICAMP, 2016).

Referências

AHMED, I.; AHMAD, M.; RODRIGUES, J. J.; JEON, G.; DIN, S. A deep learning-based social distance monitoring framework for covid-19. *Sustainable Cities and Society*, Elsevier, v. 65, p. 102571, 2021.

BAGGIO, J. V.; GONZALEZ, L. F.; BORIN, J. F. Smartparking a smart solution using deep learning. UNIVERSIDADE ESTADUAL DE CAMPINAS, 2020. Disponível em: https://smartcampus.prefeitura.unicamp.br/pub/artigos_relatorios/PFG_Joao_Victor_Estacionamento_Inteligente.pdf.

BAJAJ, A. *Performance Metrics in Machine Learning [Complete Guide]*. neptune.ai, 2022. Disponível em: https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide#:~:text=Performance%20metrics%20are%20a%20part,a%20metric%20to%20judge%20performance. Acesso em: 27 dezembro 2022.

BHATT, D.; PATEL, C.; TALSANIA, H.; PATEL, J.; VAGHELA, R.; PANDYA, S.; MODI, K.; GHAYVAT, H. Cnn variants for computer vision: History, architecture, application, challenges and future scope. *Electronics*, MDPI, v. 10, n. 20, p. 2470, 2021.

CISCO. *Cisco Annual Internet Report (2018–2023) White Paper*. Cisco, 2020. Disponível em: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. Acesso em: 26 maio 2022.

ESPíNDOLA, R.; AMORIM, N.; COUTINHO, K. *Eleitores de Nova Friburgo enfrentaram longas filas e demora para conseguir votar neste domingo*. Portal Multiplix, 2022. Disponível em: https://www.portalmultiplix.com/noticias/eleicoes-2022/confira-o-dia-de-votacao-eleitores-de-nova-friburgo-vao-as-urnas-neste-domingo-para-o-1-turno. Acesso em: 5 outubro 2022.

EVERGUARD. *Using Tech to Avoid Desensitization in Industrial Safety*. Everguard.ai, 2021. Disponível em: https://www.everguard.ai/blog/avoid-desensitization-in-industrial-safety. Acesso em: 17 agosto 2022.

FAROOQ, M. U.; WASEEM, M.; MAZHAR, S.; KHAIRI, A.; KAMAL, T. A review on internet of things (iot). *International journal of computer applications*, Foundation of Computer Science, v. 113, n. 1, p. 1–7, 2015.

G1. *Procura por vacina contra febre amarela causa fila em postos de saúde de São José*. G1, 2018. Disponível em: https://g1.globo.com/sp/vale-do-paraiba-regiao/noticia/procura-por-vacina-contra-febre-amarela-causa-filas-em-postos-de-saude-de-sao-jose-dos-campos.ghtml. Acesso em: 5 outubro 2022.

GEORGE, J.; SKARIA, S.; VARUN, V.; S., S. R. Using yolo based deep learning network for real time detection and localization of lung nodules from low dose ct scans. In: SPIE. *Medical Imaging 2018: Computer-Aided Diagnosis*. [S.l.], 2018. v. 10575, p. 347–355.

GOOGLE. *Google Colaboratory*. Google.com, s.d. Disponível em: https://colab.research.google.com/. Acesso em: 5 outubro 2022.

HASAN, M. *State of IoT 2022: Number of connected IoT devices growing 18% to 14.4 billion globally.* IoT Analytics, 2022. Disponível em: https://iot-analytics.com/number-connected-iot-devices/. Acesso em: 17 agosto 2022.

IMAGES, G. *Large group of people waiting in line*. iStock, 2011. Disponível em: https://www.istockphoto.com/br/foto/na-longa-fila-de-pessoas-esperando-gm160312819-17918341. Acesso em: 5 outubro 2022.

KUMAR, S.; TIWARI, P.; ZYMBLER, M. Internet of things is a revolutionary approach for future technology enhancement: a review. v. 6, n. 1, 2019. Disponível em: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0268-2. Acesso em: 26 maio 2022.

LAN, W.; DANG, J.; WANG, Y.; WANG, S. Pedestrian detection based on yolo network model. In: IEEE. *2018 IEEE international conference on mechatronics and automation (ICMA)*. [S.l.], 2018. p. 1547–1551.

LAROCA, R.; SEVERO, E.; ZANLORENSI, L. A.; OLIVEIRA, L. S.; GONÇALVES, G. R.; SCHWARTZ, W. R.; MENOTTI, D. A robust real-time automatic license plate recognition based on the yolo detector. In: IEEE. *2018 international joint conference on neural networks (ijcnn)*. [S.l.], 2018. p. 1–10.

LEMAY, A. Kidney recognition in ct using yolov3. *arXiv preprint arXiv:1910.01268*, 2019.

LEMOS, S. *Tutorial PyTorch: um guia rápido para você entender agora os fundamentos do PyTorch*. Insight Data Science Lab, 2022. Disponível em: https://insightlab.ufc.br/tutorial-pytorch-um-guia-rapido-para-voce-entender-agora-os-fundamentos-do-pytorch. Acesso em: 26 maio 2022.

MICROSOFT. *COCO - Common Objects in Context*. Cocodataset.org, 2015. Disponível em: https://cocodataset.org/#home. Acesso em: 26 maio 2022.

RASCHKA, S.; PATTERSON, J.; NOLET, C. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. v. 11, n. 4, p. 193, 2020. Disponível em: https://www.mdpi.com/2078-2489/11/4/193/htm. Acesso em: 26 maio 2022.

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

SFAR, A. R.; CHTOUROU, Z.; CHALLAL, Y. A systemic and cognitive vision for iot security: A case study of military live simulation and security challenges. IEEE, 2017. Disponível em: https://ieeexplore.ieee.org/document/8071828. Acesso em: 26 maio 2022.

SHAPIRO, L. G.; STOCKMAN, G. C. *Computer vision*. [S.l.]: Prentice Hall New Jersey, 2001. v. 3.

STEINKRAUS, D.; BUCK, I.; SIMARD, P. Y. Using gpus for machine learning algorithms. IEEE, 2005. Disponível em: https://ieeexplore.ieee.org/abstract/document/1575717. Acesso em: 15 agosto 2022.

TIBCO. *O que é aprendizagem supervisionada?* TIBCO Software, s.d. Disponível em: https://www.tibco.com/pt-br/reference-center/what-is-supervised-learning#:~:text=A%20aprendizagem%20supervisionada%20%C3%A9%20um,explicitamente%20programados%20para%20onde%20procurar. Acesso em: 30 junho 2022.

ULTRALYTICS. *GitHub - ultralytics/yolov5: YOLOv5 in PyTorch > ONNX > CoreML > TFLite*. GitHub, 2020. Disponível em: https://github.com/ultralytics/yolov5. Acesso em: 26 maio 2022.

UNICAMP. *Smart Campus - Unicamp*. Unicamp.br, 2016. Disponível em: https://smartcampus.prefeitura.unicamp.br/. Acesso em: 26 maio 2022.

VOXEL51. *FiftyOne — FiftyOne 0.18.0 documentation*. Voxel51.com, s.d. Disponível em: https://voxel51.com/docs/fiftyone/index.html. Acesso em: 15 novembro 2022.

YOHANANDAN, S. *mAP (mean Average Precision) might confuse you!* Medium, 2020. Disponível em: https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2. Acesso em: 15 novembro 2022.