Reconhecimento De Máscaras Utilizando Métodos De Detecção De Objetos Baseados Em Redes Neurais Convolucionais

Márcio João Ferreira

Março / 2023

Projeto de pesquisa de Mestrado em Ciência da Computação

Reconhecimento De Máscaras Utilizando Métodos De Detecção De Objetos Baseados Em Redes Neurais Convolucionais

Esse documento corresponde ao Projeto de Pesquisa apresentado à Banca Examinadora para qualificação no curso de Mestrado em Ciência da Computação do UNIFACCAMP – Centro Universitário Campo Limpo Paulista.

Campo Limpo Paulista, 31 de março de 2023.

Márcio João Ferreira

Marta Ines Velazco Fontova (Orientadora)

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Ficha catalográfica elaborada pela

Biblioteca Central da Unifaccamp

F442r

Ferreira, Márcio João

Reconhecimento de máscaras utilizando métodos de detecção de objetos baseados em redes neurais convolucionais / Márcio João Ferreira. Campo Limpo Paulista, SP: Unifaccamp, 2023.

Orientadora: Profª. Drª. Marta Ines Velazco Fontova

Dissertação (Programa de Mestrado Profissional em Ciência da Computação) – Centro Universitário Campo Limpo Paulista – Unifaccamp.

1. Algoritmos de detecção de objetos. 2. Rede neural convolucional. 3. Detecção de máscaras. I. Fontova, Marta Ines Velazco. II. Centro Universitário Campo Limpo Paulista. III. Título.

CDD – 006.32

Agradecimentos

Quero agradecer a Deus em primeiro lugar, por permitir e me dar força para a realização desse projeto de vida.

Agradeço aos meus pais: Maria Irene de Siqueira e João Inocêncio Ferreira (*in memoriam*) por todo o apoio e acreditar que eu conseguiria realizar esse sonho pessoal.

Agradeço à minha orientadora e professora Marta Ines Velazco Fontova, por todo o apoio, motivação e compreensão nessa jornada.

Ao amigo Bruno Alberto Soares Oliveira, que tive o prazer de conhecer ainda no início do projeto quando buscava por um tema em 2020. Todas as conversas e troca de experiência técnicas foram valiosas nesse trajeto de mestrado.

À amiga e ex-aluna Faccamp Elaine Barbosa de Figueiredo, pelos conselhos de como melhorar a apresentação.

A todos os professores que eu tive a honra de ser aluno e aprender, na ordem em que estudei: Marcelo, Osvaldo, Rodrigo Bonacin, Mascarenhas, Luis Mariano, Marta, Saito, Eduardo e Ana.

Agradeço ao meu empregador Sesc de Sorocaba, que me deu apoio e ajustou meus horários para que eu pudesse estar cursando as disciplinas no programa de mestrado tanto presencialmente quanto remoto.

A todos os amigos de sala que fiz durante as disciplinas, pois o papo no café e as trocas de experiências foram muito proveitosas.

A toda a equipe da Faccamp.

Resumo: *A máscara facial é tão importante quanto qualquer outro equipamento de proteção individual (EPI), pois ela diminui os riscos de elementos nocivos de serem absorvidas por vias respiratórias. A supervisão do uso de máscara exigido pelo ambiente é custosa e tediosa para o ser humano, mas os algoritmos de detecção de objetos podem colaborar nesse tipo de tarefa. Considerado o estado da arte, a aprendizagem profunda é amplamente utilizada na tarefa de detecção de objetos, que basicamente é composto por uma rede neural convolucional (Convolutional Neural Network - CNN) e um algoritmo de detecção de objetos, que deve propor as possíveis regiões candidatas de conter o objeto. A literatura apresenta o Faster R-CNN, SSD e YOLO como algoritmos relevantes na detecção objetos. As contribuições deste trabalho é um estudo específico sobre detectores de objetos na tarefa de detecção de máscaras e adiciona o critério de tempo de treinamento na avaliação. Usando um dataset de pessoas com máscaras, este trabalho compara nove modelos pré-treinados de detecção de objetos, que são: Faster R-CNN com as CNNs ResNet 50, ResNet 101, Inception-resnet-v2; SSD com as CNNs Mobilenet v2, Mobilenet v2 FPNLite e Resnet 50 FPN; e o YOLOv2, YOLOv3 e YOLOv4. Foram coletados dados quantitativos utilizando a métrica mAP, tempo de inferência e o tempo de treinamento para avaliar os modelos obtidos. Os resultados do trabalho apontam para o SSD Mobilenet v2 FPN Lite como o detector de máscara mais eficiente em termos de taxa de acerto, FPS e tempo de treinamento do modelo.*

Palavras-chave: *Algoritmos de detecção de objetos, Rede Neural convolucional, detecção de máscaras.*

Abstract: *The face mask is as important as any other personal protective equipment (PPE), as it reduces the risk of harmful elements being absorbed through the airways. Supervising the use of a mask required by the environment is costly and tedious for humans, but object detection algorithms can collaborate in this type of task. Considered the state of the art, deep learning is widely used in the object detection task, which basically consists of a Convolutional Neural Network (CNN) and an object detection algorithm, which must propose the possible candidate regions. to contain the object. The literature presents Faster R-CNN, SSD and YOLO as relevant algorithms in object detection. The contribution of this work is a specific study on object detectors in the mask detection task and adds the training time criterion in the evaluation. Using a dataset of people with masks, this work compares nine pre-trained object detection models, which are: Faster R-CNN with CNNs ResNet 50, ResNet 101, Inception-resnet-v2; SSD with Mobilenet v2, Mobilenet v2 FPNLite and Resnet 50 FPN CNNs; and YOLOv2, YOLOv3 and YOLOv4. Quantitative data were collected using the mAP metric, inference time and training time to evaluate the models obtained. The results of the work point to SSD Mobilenet v2 FPN Lite as the most efficient mask detector in terms of hit rate, FPS and model training time.*

Keywords: *Object detection algorithms, Convolutional Neural Network, mask detection.*

Sumário

Glossário

AP - *Average precision*

API - *Application Programming Interface*

BoF - *Bag of freebies*

BoS - *Bag of specials*

CNN - *Convolutional Neural Network*

CPU - *Central Processing Units*

Fast R-CNN - *Fast Regions With CNN*

Faster R-CNN - *Faster Regions With CNN*

FPN - *Feature Pyramid Network*

FPS – *Frame Per Second*

GPU - *Graphics Processing Unit*

IOU - *Intersection Over Union*

mAP - *mean Average Precision*

MTCNN - *Multi-Task Cascaded Convolutional Networks*

NMS - *Non-maximum Suppression*

R-CNN - *Regions With CNN*

RoI - *Region of Interest*

RPN - *Region Proposal Network*

SGD - *Stochastic Gradient Descent*

SSD - *Single Shot Detection*

SVM - *Support Vector Machine*

YOLO - *You Only Look Once*

Lista de Tabelas

Lista de Figuras

Capítulo 1 – Introdução

1.1 Contexto e Motivação

Existem ambientes em que a utilização de máscara é necessariamente importante na prevenção de riscos à saúde; tal dispositivo de proteção conhecido como EPI (equipamento de proteção individual) é tão importante quanto a utilização de capacete, óculos e outros (Approbato, 2021). O Instituto Butantan relata que as máscaras são sim eficazes, pois evita que as partículas, que prejudicam a saúde pelo ar, não seja absorvida pelo ser humano no ambiente em que executam suas atividades, ou quando falam ou tossem (Butantan, 2021).

No entanto, o reconhecimento de máscara ou qualquer outro objeto é algo simples e trivial para o ser humano, mas para o computador é complexo e exige alto poder de processamento (Ahmad et al., 2020). Dessa forma, a área de visão computacional vem utilizando amplamente o modelo de aprendizagem profunda (*deep learning*) para tarefas de reconhecimento de imagens; isso devido ao surgimento de grandes conjuntos de dados, o avanço das arquiteturas CNNs e as GPUs (Zhao et al., 2019).

A aprendizagem profunda é um modelo composto por uma arquitetura de rede neural multicamadas e um método de aprendizagem, comumente supervisionado. Dentre os modelos de aprendizagem profunda a CNN é o tipo de rede neural utilizada na tarefa de detecção de objetos em imagens (Zhao et al., 2019; Biswal, 2022). Além disso, as CNNs podem ser treinadas a partir de modelos pré-treinados utilizando uma técnica denominada transferência de aprendizagem (Bamne et al., 2020). Esses modelos pré-treinados são uteis na maioria dos casos em novos projetos, pois trazem os melhores conjuntos de hiper parâmetros treinados e testados em grandes conjuntos de dados.

Na Tabela 1 é apresentado os principais *datasets* de imagens utilizados para treinar CNNs em desafios de *benchmark* nas tarefas de classificação, detecção e segmentação de imagens. A primeira coluna o nome da competição, a segunda coluna o nome do *dataset*, em seguida a quantidade de imagens e a quantidade de categorias presente no *dataset*.

Desafio	*Dataset*	#Imagens	#Categorias
PASCAL VOC (Visual Object Classes) Challenge	PASCAL VOC	27,450	20
ILSVRC (ImageNet Large Scale Visual Recognition Challange)	ImageNet	Mais de 15 Milhões	1000
Microsoft Common Objects in Context Challange	MS-COCO	Mais de 200,000	80

Tabela 1 – *Datasets* em visão computacional. Fonte (Kaur and Singh, 2022).

Em relação as CNNs a literatura apresenta arquiteturas consolidadas em aprendizagem profunda. Na Tabela 2 são mostradas arquiteturas CNNs que se destacam na literatura e uma breve descrição sobre cada uma delas junto da citação.

CNN	**Descrição e citação**
Resnet	Uma CNN com blocos de conexões residuais (conexões de salto), que possibilita maior profundidade da rede neural (He et al., 2015).
Mobilenet	Uma CNN baseada em convolução separável em profundidade, que substitui uma convolução padrão por uma convolução mais barata computacionalmente (Howard et al., 2017).
Inception	Uma CNN que utiliza estratégias para aumentar a rede sem afetar complexidade computacional (Szegedy et al., 2016).
FPN	Features Pyramid Networks ou FPN é uma arquitetura independente de CNN para extração de características em escalas (Lin et al., 2017). Além disso, uma estrutura FPN com todas as camadas substituídas por convoluções separável em profundidade é chamado de FPNLite (Ghiasi et al., 2019).
YOLO	Todas as etapas de uma detecção de objetos em uma única arquitetura CNN, utilizado tanto para extração de caracteristicas quanto para detecção (Redmon et al., 2016).

Tabela 2 - Arquiteturas CNN.

Com as CNNs diversas soluções foram desenvolvidas para a detecção de máscara, por exemplo: a Resnet (Mandal, Okeukwu and Theis, 2021), Mobilenet (Sanjaya and Adi Rakhmawan, 2020), Inception (Wang, Zhao and Chen, 2021) e

YOLO (Mahurkar and Gadge, 2021).

Na Tailândia, algumas lojas adotaram a tecnologia de escaneamento de rosto para detecção de máscaras antes do cliente entrar (Hindustantimes, 2020). O trabalho de Nithyashree *et al.* (2021) propõe um sistema de detecção de máscara que elimina a necessidade de um funcionário controlar a entrada dos colaboradores no ambiente de trabalho. Sakthimani *et al.* (2021), fornece uma solução de detecção de máscara integrada com microcontroladores para abertura e fechamento de portas em prédios, que utiliza a arquitetura YOLO, Arduino e o Google *Colaboratory*.

1.2 Problemática e Justificativa

Estudos recentes exploram os algoritmos de detecção de objetos baseados em CNN para resolver desafios na detecção de máscara. Dentre os desafios estão aumentar a taxa de acerto, diminuir o custo computacional e oclusão. Por exemplo, a rede convolucional em cascata multitarefa (MTCNN) em conjunto com outras redes neurais (Facenet ou Lenet) focam em aumentar a taxa de acerto na detecção de máscara (Ejaz and Islam, 2019; Rusli et al., 2021). A CNN Mobilenet v2 criada para diminuir o custo computacional, é utilizada na detecção de máscara no trabalho de Venkateswarlu, Kakarla and Prakash, (2020). E a Retinaface, que foi desenvolvida especificamente para tentar diminuir o problema de oclusão (Hofer et al., 2021).

Para um detector de máscara eficiente, os trabalhos atuais exploram critérios como taxa de acerto, velocidade de detecção e desafios como oclusão. Porém, uma questão não abordada é o tempo de treinamento para geração do modelo, segundo Goswami (2020) um detector deve ser treinado frequentemente para manter a alta a taxa de acerto. É importante considerar o tempo da utilização dos recursos computacionais para o treinamento, principalmente se houver custo financeiro e uma alteração frequente no domínio do objeto.

Portanto, a metodologia deste trabalho visa testar nove modelos pré-treinados de arquiteturas de detecção de objetos na tarefa de detecção de máscaras e identificar o melhor detector, incluindo o tempo de treinamento na avaliação. O *dataset* usado são imagens de pessoas usando máscaras sintéticas dividido em duas partes: um conjunto de treino e um conjunto de teste. Após a conclusão do treinamento, a métrica utilizada no conjunto de teste é o mAP. Outros testes estão detalhados na metodologia deste

trabalho.

1.3 Metodologia

Conforme Figura 1, este trabalho consiste na seguinte metodologia de pesquisa:

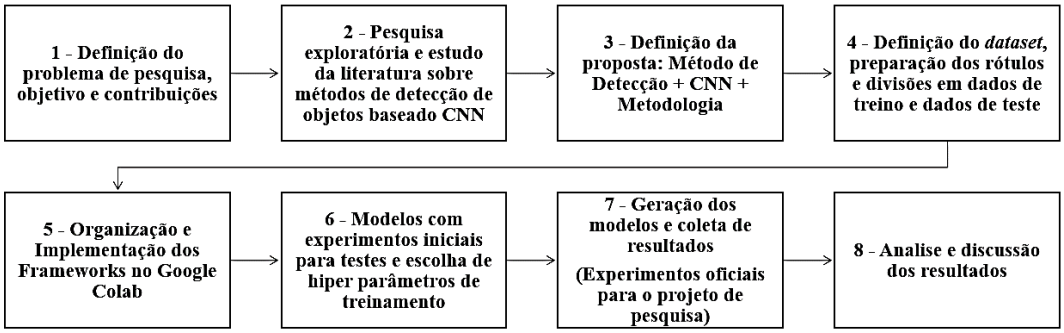


Figura 1 – Metodologia da pesquisa.

1 – A definição detalhada do problema de detecção de máscara para este projeto, identificando lacunas que possam ser discutidas para contribuir com a comunidade de aprendizado de máquina. Definição dos objetivos e contribuições;

2 – Revisão da literatura e o estudo exploratório sobre as técnicas de aprendizagem profunda em detectores de objetos aplicadas na detecção de máscara. Além disso, revisar as técnicas utilizadas em medidas de avaliação (mAP, tempo de processamento da imagem e tempo de treinamento) e *datasets*;

3 – Definição dos algoritmos de detecção de objetos para aplicar na tarefa de detecção de máscara (Faster R-CNN, SSD e YOLO). Além disso, a definição do método comparativo para a escolha do melhor detector de máscara;

4 – Definição do *dataset* (Prajna *dataset*). A rotulação da resposta da categoria em cada imagem do conjunto de dados em português brasileiro. O *dataset* está dividido em 80% para treinamento e 20% para teste. Essas imagens de teste são usadas para a gerar a área sob a curva da métrica mAP. As máscaras identificadas no *dataset* são imagens de pessoas usando máscaras sintéticas. Além disso, outras três imagens do autor serão usadas para avaliar a qualidade das detecções depois do modelo treinado e exportado para inferência, simulando um ambiente de produção.

5 – Organização dos diretórios para armazenar os arquivos de configuração, as imagens do *dataset* e os modelos obtidos. Implementar o ambiente de testes para utilização dos *frameworks*. O Google Drive será utilizado para armazenamento e o

Google Colaboratory, como ambiente de desenvolvimento.

6 – Realização de experimentos iniciais para a curva de aprendizado e testes e com isto, definir hiper parâmetros de treinamentos *batch size* e número de iterações.

7 – Realizar os treinamentos para coleta de resultados. Incialmente, foi feito o treinamento de cada modelo pré-treinado. Após isso, foi realizado a avaliação com os dados de teste, utilizando a métrica mAP@50. Para cada modelo obtido, foi feito a inferência com cinco imagens (três do autor e dois do conjunto de teste) para avaliar o tempo para processar uma imagem e a qualidade das detecções em critérios como generalização e oclusão. Por fim, foi extraído informações para obter tempo total de treinamento.

8 – Analise geral e discussão dos resultados. Os resultados obtidos foram comparados com os resultados da literatura e o estado da arte.

1.4 Objetivos

Este trabalho de pesquisa propõe-se em responder à questão: *"Qual arquitetura de detecção de objeto baseado em aprendizagem profunda é mais eficiente em taxa de acerto, processamento da imagem e tempo de treinamento, no contexto de detecção de máscaras?".*

A hipótese é que os modelos pré-treinados dos *frameworks Tensorflow API Object Detection* e *Darknet* possam contribuir com um detector de máscara de forma eficiente.

Espera-se que o modelo de detecção de máscara atenda antes de tudo uma excelente taxa de acerto. O tempo de processamento da imagem e o tempo de treinamento do detector são atributos que contribuem para uma escolha assertiva em diferentes projetos. Por exemplo, algumas aplicações podem exigir detecções robustas em aplicações em tempo real e ao mesmo tempo passar por constantes treinamentos para manter a alta taxa de acerto.

Os objetivos específicos, são:

1. Estudo dos algoritmos de detecção de objetos baseados em CNN;
2. Avaliação comparativa da taxa de acerto com a métrica mAP;

3. Avalição comparativa do tempo de processamento da imagem;
4. Qualidade das detecções considerando critérios como oclusão e generalização;
5. Avalição comparativa do tempo total para treinar um modelo.

1.5 Contribuições

As contribuições deste trabalho são:

- A adição do tempo de treinamento na avaliação do melhor detector de máscara;
- Estudo específico da literatura em algoritmos de detecção de objetos para detecção de máscara;
- Um *dataset* rotulado em português brasileiro;

Este trabalho identificou o SSD Mobilenet v2 FPNLite como o detector de objetos mais eficiente na detecção de máscara. Todos os modelos obtidos, resultados e os *datasets* estarão disponibilizados de forma publica em https://github.com/mjf2004/projeto-mestrado.

1.6 Organização do Trabalho

Logo após a introdução, os demais capítulos deste trabalho estão organizados da seguinte maneira:

- Capítulo 2 – Fundamenta os conceitos acerca de detecção de objetos e apresenta os algoritmos de detecção de objetos Faster R-CNN, SSD e YOLO, que fazem parte do referencial metodológico;
- Capítulo 3 – Revisão da literatura e trabalhos relacionados;
- Capítulo 4 – Apresenta a preparação do ambiente para os experimentos;
- Capítulo 5 – Apresenta os experimentos realizados e resultados;
- Capítulo 6 – Conclusão e os trabalhos futuros.

Encerra-se o trabalho com as referências bibliográficas e os apêndices.

Capítulo 2 – Referencial Teórico

O presente capítulo apresenta os fundamentos sobre a tarefa e os algoritmos de detecção de objetos, comumente chamados de métodos, que fazem parte do referencial teórico e metodológico deste trabalho. Na seção 2.1 apresenta a aprendizagem por transferência. Na seção 2.2 as técnicas, métricas e métodos utilizados na tarefa de detecção de objetos. Nas seções 2.3, 2.4 e 2.5 são apresentados os algoritmos de detecção de objetos Faster R-CNN, SSD e YOLO, respectivamente.

2.1 Aprendizagem por Transferência

Transferência de conhecimento ou transferência de aprendizado é a técnica utilizada por métodos de aprendizado de máquina quando o conhecimento do modelo treinado em um domínio é beneficiado em outro domínio relacionado (Pan and Yang, 2010). Por exemplo, um modelo de aprendizado de máquina que foi treinado para classificar melões e ainda não sabe o que é melancia, então basta apresentar amostras de melancias que somado ao conhecimento já adquirido sobre melão e a nova categoria será aprendida (Shao, Zhu and Li, 2015).

Em aprendizado de máquina, o conceito de transferência de aprendizado têm sido amplamente aplicado como em modelos de análise de sentimento (Arriba, Oriol and Franch, 2021), modelos de carros autônomos (Chiba and Sasaoka, 2021), classificação de imagem (Shaha and Pawar, 2018), entre outros. Segundo Pan and Yang (2010), a motivação básica para o aprendizado por transferência é evitar que o modelo seja treinado do zero com novos dados quando o espaço e a distribuição dos atributos do domínio são mudados, pois, tanto os dados quanto o esforço do treinamento do zero podem ser um processo oneroso.

Em aprendizagem profunda a rede necessita de muitos dados para evitar o *overfitting* e isso pode ser um problema, pois o pouco ajuste pela falta de dados ou o ajuste excessivo por causa de uma rede de poucas camadas, podem afetar a geração de um bom modelo. Além disso, redes com poucas camadas não conseguem capturar todas as características hierárquicas de grandes conjuntos de dados. Assim, a aprendizagem por transferência é útil e eficiente, onde determinados domínios de dados podem ser escassos, caro por serem anotados por especialistas e é um processo manual lento e propício a erros (Alzubaidi et al., 2021). Devido ao tema deste projeto, a aprendizagem

por transferência estará focada em aprendizagem profunda com as arquiteturas CNN.

2.1.1 Aprendizagem por transferência em CNN

A transferência de aprendizagem em CNN é obtida a partir de modelos pré-treinados em algum conjunto de dados, onde são carregados os pesos da CNN pré-treinado ao invés de serem inicializados aleatoriamente (Bamne et al., 2020). De acordo com Yosinski et al. (2014), a transferência de aprendizagem é eficiente por um fato comum entre as CNNs, que são os pesos das primeiras camadas não pertencer a nenhuma tarefa específica, pois são extraídas as características mais gerais.

Segundo Gluon (2022) o ajuste fino é a abordagem de transferência de aprendizagem mais comum de utilização para novos dados, pois colabora em reduzir o *overfitting* e pode generalizar melhor se o domínio do novo conjunto de dados for semelhante ao do modelo pré-treinado. No ajuste fino, o modelo pré-treinado serve para que a rede não tenha que inicializar os pesos aleatoriamente, onde as camadas iniciais são congeladas e as demais camadas são ajustadas para a nova tarefa.

Nesta dissertação é utilizada a abordagem de Ajuste Fino, usando os modelos pré-treinados para detectar objetos ajustados aos novos dados do domínio contendo pessoas utilizando máscaras. Essa abordagem cabe bem ao problema, pois os modelos pré-treinados já detectam a categoria pessoa, então basta aproveitar esse conhecimento para detectar a pessoa com máscara.

2.2 Detecção de Objetos

A tarefa de detectar objetos é definida como localizar um ou mais instâncias de um objeto na imagem. Geralmente, essa localização é dada por coordenadas de caixas delimitadoras retangulares acompanhadas do objeto e uma pontuação de confiança. Além disso, detectores de objetos de última geração estão formados por: uma CNN para extração de características na imagem e o método (algoritmo) de detecção de objetos para determinar as possíveis regiões que podem conter o objeto (Jiao et al., 2019; Bochkovskiy, Wang and Liao, 2020). A seguir são apresentados os componentes acerca de detectores de objetos como a métrica intersecção sobre união, pós-processamento, âncoras, as etapas de detecção de objetos e as métricas de avaliação.

2.2.1 Aumento de dados

A técnica conhecida como aumento de dados (*Data Augmentation*) cria dados artificialmente a partir de uma imagem de origem. A técnica colabora para que o modelo não fique super ajustado aos poucos dados existentes, problema recorrente em modelos de aprendizagem profunda (Goyal, 2019). As transformações mais comuns são: transformações geométricas, inversão, rotação, translação e injeção de ruído (Shorten and Khoshgoftaar, 2019). A Figura 2 mostra a técnica aplicada em uma imagem de entrada criando seis amostras com diferentes transformações.

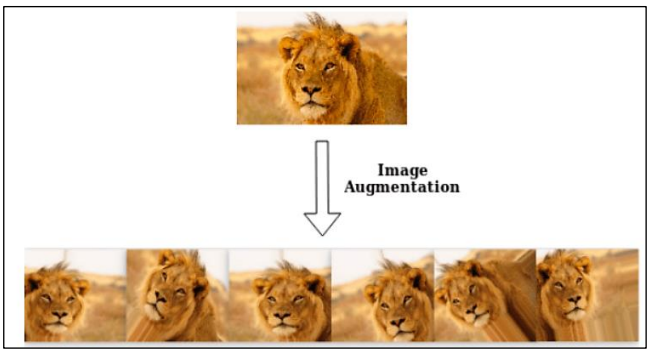


Figura 2 - Técnica Aumento de dados. Fonte (Goyal, 2019).

2.2.2 *Intersection over union*

A detecção pode ser medida através da métrica intersecção sobre união (*intersection over union* – IOU) ou similaridade de Jaccard. O cálculo é realizado utilizando uma caixa delimitadora prevista pelo modelo com uma caixa resposta anotada na imagem do *dataset*. A Equação 1 mostra o cálculo entre a área de duas caixas delimitadoras, uma caixa prevista e uma caixa resposta (Padilla, Netto and Silva, 2020).

$$J\left(C_{previsto}, C_{resposta}\right) = \frac{|\,C_{previsto} \cap C_{resposta}|}{|C_{previsto} \cup C_{resposta}|} \therefore \text{IOU} = \frac{\acute{A}rea\ da\ Intersecção}{\acute{A}rea\ da\ Uni\tilde{a}o} \qquad (1)$$

Os pipelines dos detectores de objetos fazem uso da IOU para obter informações como: determinar a pontuação de confiança de conter o objeto, para avaliar um modelo de detecção e na fase de pós processamento (Redmon et al., 2016; Bodla et al., 2017; Padilla et al., 2021). A Figura 3 mostra exemplos de sobreposição de caixas de caixas previstas (vermelho) e caixas de respostas (verde) com as respectivos valores IOU.

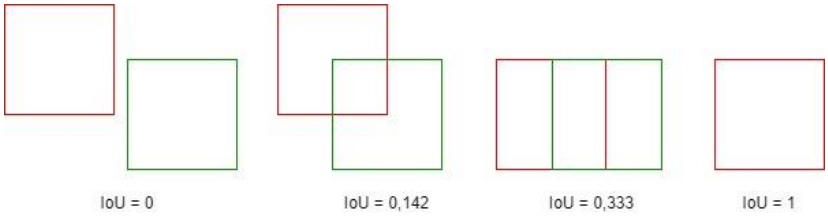


Figura 3 - Intersecção sobre união. Fonte (baeldung, 2022).

2.2.3 *Non-maximum suppression*

É comum na etapa de pós-processamento o detector de objeto utilizar uma abordagem para reduzir o número de caixas delimitadoras até que se tenha uma caixa por objeto. O método mais utilizado nessa tarefa é a supressão não máxima (*non-maximum suppression* – NMS). Este método utiliza a caixa de maior pontuação e remove as caixas vizinhas de menor pontuação de forma recursiva. O problema do NMS é que as outras caixas delimitadoras mesmo dentro do limiar determinado serão perdidas. No entanto, outra abordagem que decai a pontuação de todas as outras detecções durante as iterações e contorna esse problema é chamado de Soft-NMS (Bodla et al., 2017; Jiao et al., 2019).

A Figura 4 mostra a execução do método NMS em um detector de objetos. Inicialmente uma imagem de entrada é fornecida (a), em seguida inúmeras caixas delimitadoras são geradas por objeto (b), na sequência a fase de pós-processamento executa recursivamente o método NMS (c) e finalmente a imagem de saída é apresentada com apenas uma caixa delimitadora para cada instância do objeto localizada na imagem de entrada (d).

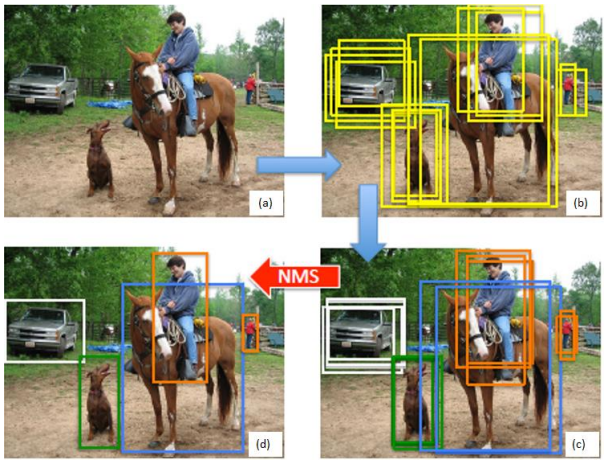


Figura 4 – Execução do método NMS. Fonte (Bodla et al., 2017).

2.2.4 Âncoras

Âncoras são caixas delimitadoras que propõe as possíveis regiões onde se localizam os objetos na imagem, presentes em métodos de detecção como Faster R-CNN, SSD e YOLO. As âncoras podem prever o local e o tamanho dos objetos em tempo ágil mesmo que os objetos tenham tamanhos diferentes (Liu et al., 2016; Ren et al., 2016; Mathworks, 2022).

A cada âncora contém informações como posição, tamanho e a probabilidade de um determinado objeto. Além disso, o tamanho dos objetos contidos no conjunto de dados de treinamento geralmente define a escolha das características das âncoras para a detecção (Redmon and Farhadi, 2017; Mathworks, 2022).

As âncoras não sabem exatamente o local e o tamanho de cada objeto e é durante o treinamento que aprendem a corrigir os deslocamentos necessários (Mathworks, 2022). Para isso, são utilizadas as sobreposições entre as caixas delimitadoras de âncora e as caixas delimitadoras com as respostas. No processo, a caixa de âncora de maior valor IOU é usada como referência para ser aprimorada. Portanto, as âncoras são treinadas para se tornarem especializados em cada tipo de objeto, com tamanho, forma e localização (Christiansen, 2018).

2.2.5 Etapas da detecção de objetos

O processo de detecção de objetos tem quatro etapas: pré-processamento, extração de características, detecção e pós-processamento (Jiao et al., 2019; Bochkovskiy, Wang and Liao, 2020). Cada etapa é descrita a seguir:

- Pré-processamento – é a etapa onde as imagens são redimensionadas para corresponder a entrada da CNN. Nesta etapa, o aumento de dados pode ser aplicado para alimentar a rede com diversas versões da imagem do original;
- Extração de características – são gerados os mapas de características, que influenciam na qualidade das detecções na etapa seguinte. Geralmente, o *backbone* de um detector de objetos é uma rede pré-treinado em um domínio específico. Ainda pode existir uma etapa intermediária entre o *backbone* e a etapa de detecção, denominada *neck*, que coleta mapas de características em diferentes estágios de escala, o FPN por exemplo.
- Detecção – um algoritmo de detecção de objetos é aplicado no mapa de característica, que propõe e ajusta as caixas delimitadoras com a pontuação de confiança do objeto localizado. Alguns algoritmos de detecção conhecidos são RPN, SSD, YOLO e Faster R-CNN;
- Pós-processamento – esta etapa exclui todas as caixas delimitadoras com pontuação fraca e reduz as detecções a uma única caixa, o NMS por exemplo.

2.2.6 Medidas de avaliação

Inicialmente para avaliar um modelo de detecção de objetos é necessário definir um valor de confiança entre uma caixa prevista (âncora) e uma caixa de resposta, e isso é obtido através da métrica IOU. Assumindo um limite (t) de confiança, temos as definições:

- Verdadeiro Positivo (*True Positive* - TP) – classificado como uma detecção correta quando a caixa prevista tem confiança igual ou maior que o limite (t);
- Falso Positivo (*False Positive* - FP) – classificado como uma detecção incorreta quando a confiança é menor que o limite (t);

- Falso Negativo (*False Negative* - FN) – a anotação está presente na imagem e não foram geradas caixas previstas desse objeto;
- Verdadeiro Negativo (*True Negative* - TN) – são todos os objetos não anotados na imagem e que não tiveram caixas previstas. Não se aplica em detectores de objetos.

Segundo Everingham *et al.* (2015), os detectores de objetos podem apresentar mais de uma sobreposição VP para uma resposta, nesse caso apenas uma, a de maior sobreposição é considerada VP e as demais são consideradas FP.

As medidas de avaliação comumente utilizadas em tarefas de aprendizado de máquina, são: acurácia, precisão, revocação, pontuação F1 e precisão média (Filho, 2018). Descritas a seguir.

- Acurácia (*Accuracy*) – acurácia é uma métrica de taxa de acerto para avaliar tarefas de classificação e é obtida através da proporção entre a soma de verdadeiros positivos e verdadeiros negativos sobre todos os casos, representado por [(TP + TN) / (TP + TN + FP + FN)] (Nagrath et al., 2021);
- Precisão (*Precision*) – é o percentual dos verdadeiros positivos para o total de todas as detecções previstas pelo modelo, representado por (VP / VP + FP) ou VP / todas as detecções. Em detectores de objetos é o percentual de verdadeiros positivos detectados;
- Revocação (*Recall*) – é o percentual dos verdadeiros positivos para o total de caixas anotadas na imagem, representado por (VP / VP + FN) ou VP / todas as regiões anotadas. Em detectores de objetos é a capacidade de detectar todas as regiões anotadas;
- Pontuação F1 (F1 *Score*) – é uma média ponderada da precisão e da revocação, representado por (Precisão × Revocação) / [(Precisão + Revocação) / 2];
- Precisão Média (*Average Precision* – AP) – embora chamada de precisão média, essa métrica é um valor escalar interpretado pela área sob a curva do gráfico da precisão (eixo y) e revocação (eixo x) plotados em diferentes pontos de detecções. Além disso, para facilitar o cálculo é

utilizado alguma abordagem de interpolação (Arcgis, 2022).

Área sob a curva de precisão × revocação (AP)

A área sob a curva ou AP fornece o valor de avaliação de apenas um objeto. Um bom detector deve detectar todas as anotações na imagem (FN = 0) enquanto detecta corretamente todas as anotações (FP = 0), isso fornece a maior área possível independentemente do valor de confiança (*t*) determinado. Por exemplo, no desafio PASCAL VOC é adotado IOU de 50% , chamado de AP@50 (Padilla, Netto and Silva, 2020).

A média de todos os APs é conhecido como *mean Average Precision* ou mAP e é a métrica comumente usada em detectores de objetos (Padilla, Netto and Silva, 2020). Além disso, alguns desafios em detecção de objetos não fazem distinção entre AP ou mAP, a interpretação é dada pelo contexto (COCO, 2022).

Um exemplo mostrado na Figura 5 mostra o cálculo do objeto gato sobre 12 pontos de detecção utilizando IOU de 50%. Para formar a área, os valores de confiança das caixas delimitadoras previstas são organizados em ordem decrescente com seu respectivo cálculo de precisão e revocação. Geralmente, a área formada é de difícil cálculo e por isso são usadas técnicas de interpolação, como a interpolação de todos os pontos (Padilla et al., 2021).

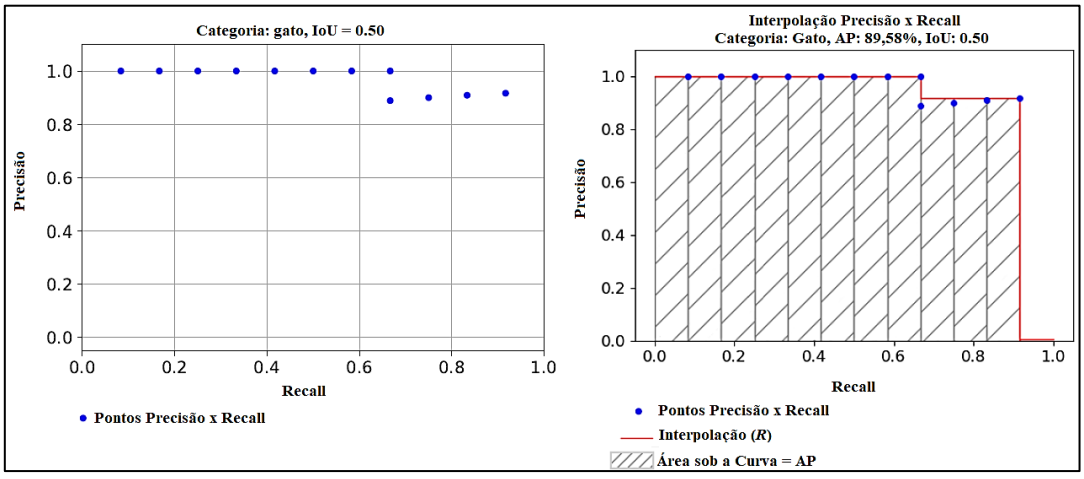


Figura 5 – Interpolação de todos os pontos. Fonte (Padilla et al., 2021).

2.3 Faster R-CNN

Faster R-CNN é um método para detecção de objetos de dois estágios. No primeiro estágio gera-se as propostas de regiões candidatas ou caixas delimitadoras, com uma rede neural chamada de RPN e, em seguida calcula-se as previsões. Este método é uma evolução dos antecessores R-CNN e Fast R-CNN.

O R-CNN (*Regions with* CNN) é o primeiro algoritmo para detecção de objetos baseado em CNN e que trouxe contribuições de desempenho superior a outras abordagens (Jiao et al., 2019).

Conforme Figura 6, o R-CNN utiliza um algoritmo chamado de busca seletiva para gerar aproximadamente 2.000 caixas delimitadoras em uma imagem de entrada. A largura e a altura de cada proposta de região são reformatadas e em seguida, enviada para uma CNN. A saída da CNN gera um vetor de dimensão fixa, que é enviado para um classificar SVM (*support Vector Machine*). As caixas delimitadoras propostas são ajustadas através da regressão das coordenadas (Girshick et al., 2014).

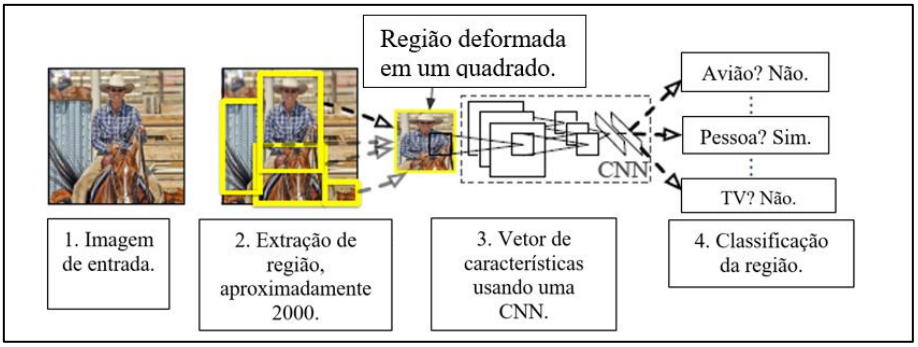


Figura 6 – R-CNN. Fonte (Girshick et al., 2014).

No ano seguinte surge o Fast R-CNN que agregou os três estágios da versão anterior (CNN, SVM e regressão de caixas delimitadoras) em uma única arquitetura, aumentando o poder de computação compartilhada.

O Fast R-CNN evita que a CNN execute 2000 vezes a extração de características. A imagem de entrada passa uma única vez pela CNN, que gera inúmeras propostas de regiões de interesse (*RoI*) da última camada de convolução (por busca seletiva). As propostas de regiões de interesse são agrupadas em uma camada de *pooling* (*RoI pooling*) e são enviadas uma por vez para as camadas totalmente conectadas. Cada *RoI* é redimensionado para um vetor de tamanho fixo, que é utilizado

para classificação (nesta versão as SVMs foram substituída pela *softmax*) e regressão (Girshick, 2015; Hui, 2017).

RPN

Algoritmos para geração de propostas de região como, busca exaustiva ou busca seletiva, são lentos para um sistema de detecção. A busca exaustiva utiliza janelas de tamanho e proporção fixa e é impraticável procurar objetos em todos os lugares da imagem. A busca seletiva utiliza a busca exaustiva e a segmentação de pixels com diferentes métricas como: cor, textura, escala. Logo, o tempo gasto com geração de propostas torna-se um desafio para algoritmos de detecção de objetos de alto desempenho (Uijlings et al., 2013; Ren et al., 2016).

O Faster R-CNN supre a deficiência dos anteriores com uma rede para proposta de região (*Region Proposal Networks* - RPN) em substituição do algoritmo de busca seletiva. A abordagem do método com a RPN torna a solução eficiente e o custo computacional baixo, devido ao compartilhamento de camadas convolucionais (Ren et al., 2016).

A RPN é uma rede totalmente convolucional que gera as propostas de regiões em cada local espacial do último mapa de características. E, através de uma janela deslizante tenta capturar os objetos com caixas de âncoras de tamanho pré-determinado (Ren et al., 2016).

Através de uma janela deslizante uma pequena rede tenta capturar os objetos com caixas de âncoras de tamanho pré-determinado. O número de âncoras é definido por *k* e cada âncora contém dois neurônios para classificação de objetividade (objeto ou fundo da imagem) e quatro neurônios para regressão das coordenadas da caixa delimitadora. A rede aprende a prever o deslocamento das caixas de âncoras através das respostas anotadas na imagem. As caixas de âncoras por padrão utilizam três proporções e três escalas em cada local no mapa de características (Ren et al., 2016).

Logo após do método NMS, as propostas de regiões obtidas através da janela deslizante são agrupadas em uma camada *de* agrupamento de regiões. O mapeamento das propostas regiões e *pooling* é compartilhado com o mapa de características da última camada convolucional e em seguida essas propostas são enviadas cada uma por vez para o processamento de classificação e regressão em uma Fast R-CNN, que

compartilha o mesmo *backbone* (Ren et al., 2016; Santos, 2020).

Algoritmo Faster R-CNN:

- 1º estágio:
  - A imagem de entrada passa através da CNN de *backbone* e obtém o mapa de características;
  - Passa o mapa de características para RPN obter as propostas de região através de *k* âncoras;
- 2º estágio:
  - É executado NMS e as saídas são as propostas de região são agrupadas (*RoI pooling*) e cada uma é redimensionada para um tamanho fixo;
  - Passa as propostas redimensionadas para a FC prever a classificação final e os ajustes da caixa delimitadora.

2.4 SSD

O detector de disparo único (*Single Shot Detector* - SSD) é um método para detecção de objetos, que gera detecções em um único estágio (*Liu et al.* 2016). O detector prevê as pontuações de confiança e a regressão de caixas delimitadoras a partir de um conjunto de âncoras em sucessivos mapas de características de diferentes escalas. Diferente do Faster R-CNN, este método não tem a etapa de geração de propostas separada. O SSD combina as diversas previsões para tratar diferentes tamanhos de objetos, isso melhora de forma significativa a velocidade mantendo uma precisão competitiva.

Uma parte fundamental da estrutura do SSD é a extração do mapa de características com diferentes escalas. Em cada local espacial do mapa é analisado um conjunto de âncoras com tamanhos distintos para o treinamento do modelo (Liu et al., 2016).

Um detector com o método de detecção SSD contém duas partes: o *backbone* para extração de características (rede base) e o método de detecção SSD (cabeça de detecção). As previsões são realizadas com filtros convolucionais em mapas de características auxiliares adicionadas ao final do *backbone*, que diminuem

gradativamente com a intenção de detectar objetos menores (Liu et al., 2016; ArcGIS, 2021).

As âncoras são combinadas com qualquer resposta de IoU superior a um limiar (*t*), dispensando a escolha da sobreposição de maior valor. As correspondências negativas são classificadas usando a confiança de maior valor mantendo a proporção de três previsões negativas para uma previsão positiva. Essa estratégia, denominada *hard negative mining*, é utilizada para estabilizar o treinamento (Liu et al., 2016).

Algoritmo SSD:

- Imagem de entrada para a extração de características com a rede base;
- Gera-se um conjunto fixo de âncoras usando filtros de convolução 3 × 3 para cada célula (localização) do mapa e assim as previsões (cada filtro gera [C + 4] de canais de profundidade como saída);
- O SSD é aplicado em 6 camadas auxiliares de forma independente com resolução e dimensão que decrementam progressivamente permitindo detecções multiescala;
- NMS para a detecção final.

2.5 YOLO

Você apenas olha uma vez (*You Only Look Once* - YOLO) é um algoritmo para detecção de objetos de um estágio (Redmon *et al*., 2016).

O método prevê ao mesmo tempo a caixa delimitadora e a probabilidade do objeto em uma única rede neural convolucional (Redmon et al., 2016). A Figura 7 mostra as etapas do YOLO, onde a) recebe uma imagem de entrada dividida em $S \times S$ células, b) para cada célula é previsto B caixas delimitadoras com pontuação de confiança e paralelamente a probabilidade do objeto e em c) o NMS é usado para mostrar a detecção final.

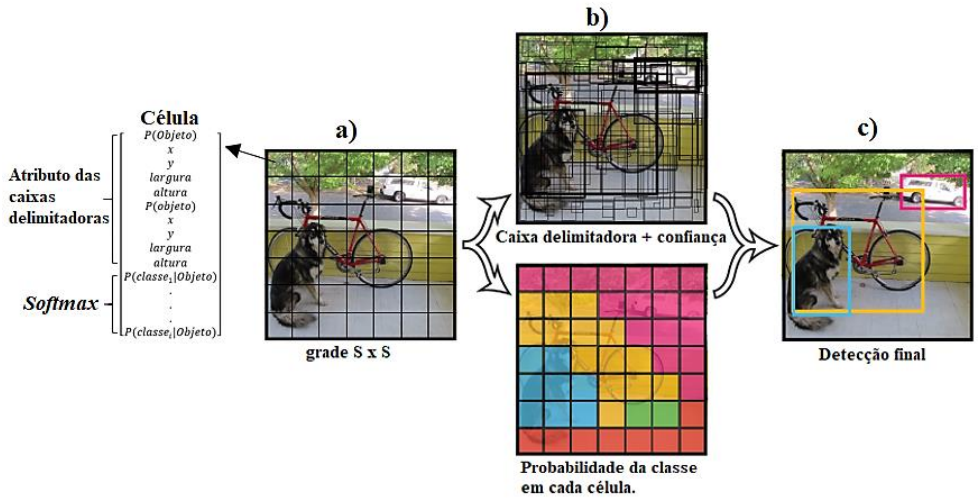


Figura 7 - Visão Geral YOLO. Fonte (Kamal, 2021).

YOLOv2

A primeira versão YOLO detecta apenas um objeto por célula da grade. Isso limita o método a não detectar mais de um objeto, caso eles estejam na mesma célula. Esse problema foi contornado no YOLOv2 com a utilização de *k* âncoras em cada célula da grade. As âncoras são geradas diretamente na camada convolucional e a âncora de maior valor IOU é atribuída ao objeto (Redmon and Farhadi, 2017; Kamal, 2021).

O YOLOv2 usa um algoritmo de agrupamento (*k-means*) para gerar o tamanho das caixas de âncoras, ao invés de valores pré-definidos manualmente como no Faster R-CNN e SSD. Cada caixa de âncora gera predições para as coordenadas da caixa delimitadora (t_x, t_y, t_w e t_h) e a probabilidade de conter o objeto (Redmon and Farhadi, 2017).

Conforme figura 8, o YOLOv2 gera a localização das coordenadas da âncora (caixa pontilhada) em relação a célula da grade. Os valores aleatórios gerados para as coordenadas do centro da âncora podem trazer instabilidade nas iterações iniciais no processo de treinamento. Para evitar que isso aconteça a função sigmoide é utilizada para limitar os valores previstos da caixa de âncora no intervalo [0,1], representado por $\sigma(t_x)$ e $\sigma(t_y)$, que somado ao deslocamento c_x *e* c_y define o centro da caixa delimitadora final (caixa azul) (Redmon and Farhadi, 2017).

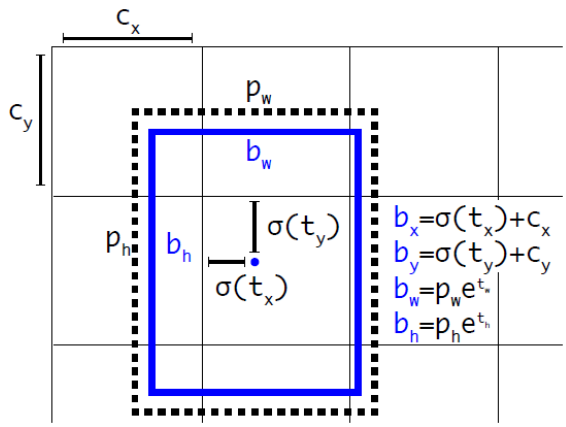


Figura 8 – Coordenadas de âncoras YOLO. Fonte (Redmon and Farhadi, 2017).

YOLOv2 trouxe outras melhorias de desempenho em uma rede totalmente convolucional (Redmon and Farhadi, 2017). As melhorias são apresentadas a seguir:

- *Batch normalization* – é introduzida para estabilizar o treinamento e acelerar a convergência e regularizar o modelo;
- Classificador de alta resolução – No YOLOv1 o classificador é pré-treinado em resolução 224×224 e depois alterna a entrada para 448×488 na tarefa de detecção. Isso tem um custo computacional para a rede se ajustar à nova resolução de entrada. O YOLOv2 pré-treina o classificador na resolução 224×224 e na sequência, o classificador é treinado por 10 épocas com a resolução 448×448. O resultado desse processo é uma melhor adaptação de ajuste com os filtros de detecção em alta resolução. O mAP foi incrementado em 4% com essa estratégia no COCO *dataset*.
- Imagem de entrada – o pré-processamento que redimensiona a imagem de entrada de 448×448 do YOLOv1 para 416×416 no YOLOv2 (sem a camada de *pooling*) produz um mapa de características de dimensão espacial ímpar. Essa alteração implica que objetos maiores tendem a ocupar o centro da imagem e o número ímpar de células facilita determinar a qual célula o objeto pertence.

YOLOv3

A principal dificuldade do YOLOv2 é a detecção de objetos pequenos. O YOLOv3 é uma atualização, que melhorou o desempenho de detecção com a predição

multi escala. A nova CNN, chamado de Darknet-53, utiliza conexões de atalho (como na Resnet) e isso permite que a rede tenha mais profundidade que a versão anterior. As detecções são realizadas em três escalas diferentes com um filtro de detecção aplicado em determinadas camadas da rede.

Outra melhoria do YOLOv3 é a classificação multi rótulo. Para fazer a previsão de categoria multi rótulo, em cada caixa delimitadora é utilizado um classificador logístico independente no lugar da função softmax. Isso não afeta o desempenho e evita a sobreposição de categorias no *dataset*, por exemplo: mulher e pessoa. (Redmon and Farhadi, 2018).

YOLOv4

Considerado o estado da arte na tarefa em detecção de objetos, até o momento da revisão da literatura deste projeto, o YOLOv4 é um aperfeiçoamento da versão anterior pulicada por (Bochkovskiy, Wang and Liao (2020), que consiste em três componentes principais especializados em detecção, que são: Uma CNN (CSP-Darkenet-53), uma camada para extração de características em escala e o algoritmo para detecção de objetos (YOLOv3).

O YOLOv4 adota duas categorias de otimizações que melhoram a detecção de objetos, chamados de *Bag of freebies* (BoF) e *Bag of specials* (BoS). O *Bag of freebies* é um conjunto de estratégias aplicado durante o treinamento para melhorar a precisão e o *Bag of specials* são módulos de pós-processamento que melhoram a precisão, com um certo custo de inferência (Wang et al., 2019; Bochkovskiy, Wang and Liao, 2020).

Como pode ser visto na Figura 9, treinado no *dataset* COCO com uma GPU Tesla V100, o YOLOv4 tem um desempenho considerável em termos de velocidade e taxa de acerto em relação ao YOLOv3. A taxa de acerto e velocidade melhorou 10% e 12%, respectivamente no YOLOv4.

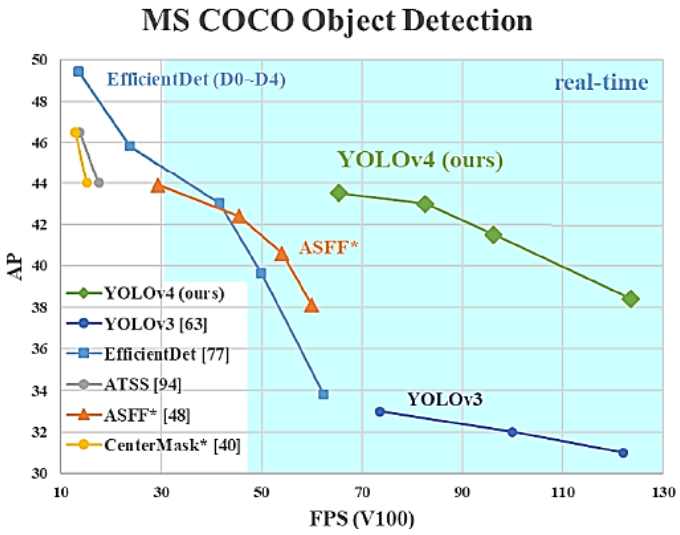


Figura 9 – Desempenho YOLOv4. Fonte (Bochkovskiy, Wang and Liao, 2020).

Algoritmo YOLO:

- A imagem de entrada é dividida em uma grade $S \times S$ células, onde S é uma célula da grade;
- Para cada célula da grade é feito a previsão de B caixas delimitadoras (âncoras);
- Para cada âncora, a célula prevê as coordenadas de localização, probabilidade e confiança do objeto;
- NMS para a detecção final.

2.6 Considerações Finais

Os métodos de detecção Faster R-CNN, SSD e YOLO apresentados nesse capítulo fazem parte do referencial metodológico deste trabalho. Eles foram usados nos experimentos e os resultados foram analisados com o intuito de identificar um melhor detector de máscara nos requisitos taxa de acerto, tempo de inferência e tempo de treinamento.

No próximo capítulo será mostrado a revisão da literatura e como foi identificado os algoritmos de detecção de objetos mais relevantes para ser usado neste trabalho. A revisão inclui os *datasets* na tarefa de detecção de máscara, os métodos de avaliação e resultados como trabalhos relacionados.

Capítulo 3 – Revisão da Literatura e Trabalhos Relacionados

O presente capítulo aborda a revisão da literatura. Os procedimentos de revisão sistemática foram baseados em Martins (2018). As etapas de revisão consistem no planejamento e execução, a discussão, os resultados são apresentados como trabalhos relacionados e as finalizando com as considerações finais.

3.1 Execução da Revisão

A revisão de literatura apresentada neste capítulo tem como direcionamento o trabalho de Jiao et al. (2019), que destaca os algoritmos de detecção de objetos baseados em aprendizagem profunda como o estado da arte nesse tipo de tarefa. Ainda explora outros aspectos na tarefa de detecção de objetos como arquitetura CNNs e *datasets*. Os métodos de detecção mais relevantes pelo autor foram a detecção em dois estágios Faster R-CNN e a detecção em um único estágio SSD e YOLO.

Com base nas considerações levantadas pelo autor, este trabalho considerará esses três métodos de detecção de objetos. Portanto, a pesquisa parte da seguinte questão investigativa: “quais modelos de aprendizagem profunda usando o Faster R-CNN, SSD ou YOLO foram utilizados no desenvolvimento de soluções para detecção de máscara?”.

O período de pesquisa e levantamento de artigos foi entre junho-julho/2021 e uma atualização de trabalhos relevantes em janeiro/2022. A Tabela 3 mostra os critérios de busca dos artigos, a primeira linha representa o intervalo das publicações, a segunda linha as bases cientificas, a terceira linha as palavras-chave e a última linha o idioma das publicações.

Período:	2020-2022
Bases de periódicos:	Science Direct, Springer Link, IEEE Xplore, ACM Digital Library, Arxiv
Palavras-chave:	*Recognition*, *Mask*, *Face*, *Detect*, Covid-19
Texto da Busca:	(*“face oumask”*) *e* (*“detect”* ou *“recognition”* ou *“covid-19”*)
Idioma:	Inglês

Tabela 3 – Critérios de Busca.

Os artigos relacionados incluídos na discussão deste trabalho atendem os seguintes critérios:

a) Artigos que abordem a detecção de máscara facial;

b) Artigos que tenha como base da solução alguma arquitetura CNN;
c) Artigos que utilizem os algoritmos de detecção de objetos Faster R-CNN, SSD ou YOLO na solução;
d) Artigos de bases cientificas e artigos das bases cientificas sem revisão por pares (Arxiv) nos últimos três anos;
e) Os artigos que tratam soluções para além da detecção de máscara como distância entre pessoas, temperatura corporal e reconhecimento facial foi considerado apenas a técnica utilizada na detecção de máscara;
f) Estudos do tipo *survey* sobre detecção de máscara.

Foram excluídos os artigos com os seguintes critérios:

a) Excluídos artigos que não tenham na proposta de detecção os métodos de detecção Faster R-CNN, SSD ou YOLO;
b) Foram excluídos os artigos com mais de três anos;
c) Excluídos os artigos que não puderam ser lidos integralmente para análise das propostas, com exceção dos artigos do tipo *survey*;
d) Excluídos os artigos sem medida de avaliação clara e definida.

Conforme mostra a Tabela 4, a revisão da literatura obteve sessenta e cinco artigos (65) levantados entre as bases cientificas: Science Direct, Springer Link, IEEE Xplore, ACM Digital Library e Arxiv. A aplicação do critério de exclusão “b” removeu um (1) artigo, o critério de exclusão “c” foi removido nove (9) artigos e o critério de exclusão “d” três (3) artigos, totalizando 52 artigos para serem lidos e analisados. Após análise, foi notado que os métodos de detecção Faster R-CNN, SSD e YOLO não faziam parte da proposta do artigo, portanto foram removidos conforme critério “a”, resultando em vinte e dois (22) artigos.

<table>
<tr><th colspan="2">Levantamento
Total: 65 artigos</th><th colspan="2">Analisados
Total: 52 artigos</th><th colspan="3">Selecionados
Total: 22 artigos</th></tr>
<tr><th>Bases cientificas</th><th>#</th><td rowspan="7">Dos 65 artigos, 13 artigos foram removidos, conforme critérios de exclusão b, c e d. Resultando em 52 a serem analisados.</td><th>#</th><td rowspan="7">Dos 52 artigos.
30 foram removidos, conforme critério de exclusão “a”.</td><th>#</th><th>%</th></tr>
<tr><td>Science Direct</td><td>8</td><td>7</td><td>4</td><td>18%</td></tr>
<tr><td>Springer Link</td><td>18</td><td>14</td><td>4</td><td>18%</td></tr>
<tr><td>IEEE Xplore</td><td>20</td><td>13</td><td>7</td><td>32%</td></tr>
<tr><td>ACM Digital Library</td><td>6</td><td>5</td><td>3</td><td>14%</td></tr>
<tr><td>Arxiv</td><td>13</td><td>13</td><td>4</td><td>18%</td></tr>
<tr><td>Total</td><td>65</td><td>Total</td><td>52</td><td>Total</td><td>22</td><td>100%</td></tr>
</table>

Tabela 4 – Síntese de artigos selecionados.

Foram retirados dos artigos selecionados características relevantes como:

- *Dataset*, arquitetura CNN e o algoritmo de detecção;
- Se foi utilizado transferência de aprendizagem;
- O tipo de método de avaliação proposta, categorizadas neste trabalho como:
  - Aplicação – um detector aplicado em um *dataset* para coleta de resultados, com uma medida de avaliação definida;
  - Comparativo – Comparativo entre detectores de máscara em um *dataset* com uma medida de avaliação definida;
  - *Survey* – Estudo de revisão sobre as técnicas utilizadas em detecção de máscara;
- Medida de avaliação.

A revisão realizou as seguintes etapas: a identificação dos algoritmos de detecção de objetos; o levantamento de trabalhos relacionados na tarefa de detecção de máscara (conforme critérios de busca); leitura e análise das propostas para extração de informações relevantes; seleção de estudos conforme critérios de inclusão e exclusão; síntese de estudos selecionados (conforme Tabela 3).

3.2 Resultados da Revisão

A Tabela 5, a seguir, mostra a lista de trabalhos selecionados na revisão da literatura. As informações extraídas desses estudos foram categorizadas nas seguintes colunas: identificação, título e a citação, o método de detecção que aparece na metodologia, *dataset*, medida de avaliação, *transfer learning* e o tipo de método aplicado nos experimentos (aplicação, comparativo ou *survey*).

Id	Título e citação	Método de detecção	*Dataset*	Medida de Avaliação	Transfer Learning	Método
A1	*FMD-Yolo: naefficient face mask detection method for COVID-19 prevention and control in public.* (Wu et al., 2022)	YOLOv3 e Faster R-CNN	Kaggle e *Dataset* Customizado	mAP	Sim	Comparativo
A2	*SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNet v2*. (Nagrath et al., 2021)	SSD	Kaggle e Prajana	*F1-score* e *Accuracy*	Sim	Comparativo
A3	*IoT-Enabled smartdoors for monitoring body temperature and face mask Detection.* (Varshini et al., 2021)	SSD	Prajna	*F1-score* e *Accuracy*	Sim	Aplicação
A4	*Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask Detection.* (Loey et al., 2021)	YOLOv2	MMD e Kaggle	mAP	Não	Comparativo
A5	*MobileNet Mask: A Multi-phase Face Mask Detection Model toPrevent Person-To-Person Transmission of SARS-CoV-2.* (Dey, Howlader and Deb, 2021)	SSD	Kaggle, RMFD e Prajna	*Accuracy, F1-Score, Precision e Recall*	Sim	Aplicação
A6	*Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment.* (Singh et al., 2021)	YOLOv3 e Faster R-CNN	Mafa e WiderFace	mAP	Sim	Comparativo
A7	*A hybridtiny YOLO v4-SPP module based improved face mask detection vision system.* (Kumar et al., 2021)	YOLOv2, YOLOv3 e YOLOv4	*Dataset* Customizado	mAP	Sim	Comparativo

A8	*Face mask detection in COVID-19: a strategic review.* (Vibhuti et al., 2022)	Survey	*Survey*	*Survey*	*Survey*	*Survey*
A9	*COVID-19 Monitoring System using Social Distancing and Face Mask Detection on Surveillance video datasets.* (Srinivasan et al., 2021)	YOLOv3	*Dataset* Customizado	*Accuracy, F1-Score, Precision e Recall*	Não	Comparativo
A10	*naApplication of Deep-Learning Techniques to Face Mask Detection During the COVID-19 Pandemic.* (Khamlae, Sookhanaphibarn and Choensawat, 2021)	YOLOv5	Kaggle	mAP	Sim	Aplicação
A11	*Real-Time Face Mask Detector Using YOLOv3 Algorithm and Haar Cascade Classifier*. (Vinh and Anh, 2020)	YOLOv3	Mafa	*Precision* e *Recall*	Não	Comparativo
A12	*Hybrid Transfer Learning and Broad Learning System for WearingMask Detection in the COVID-19 Era.* (Wang, Zhao and Chen, 2021)	Faster R-CNN e SSD	WMD	*Precision, Recall e F1-score*	Sim	Comparativo
A13	*A Brief Review of Convolutional Neural Network Techniques for Masked Face Recognition.* (Ibitoye, 2021)	*Survey*	*Survey*	*Survey*	*Survey*	*Survey*
A14	*Mask wearing detection method based on SSD-Maskalgorithm.* (Xu et al., 2020)	SSD	WiderFace Mafa	*mAP*	Não	Aplicação
A15	*A Survey on Masked Facial Detection Methods and Datasets for Fighting Against COVID-19.* (Wang, Zheng and Chen, 2022)	*Survey*	*Survey*	*Survey*	*Survey*	*Survey*
A16	*Deep Learning-based Face Mask Detection Using Automated GUI for COVID-19*. (Bhattarai, RajPandeya and Lee, 2021)	Faster R-CNN	MASKEDFACE-NET	*F1-score*, mAP	Sim	Comparativo
A17	*Face Mask Detection on Real-World Webcam Images*. (Adhikarla and Davison, 2021)	YOLOv3 e Faster R-CNN	*Dataset* Customizado	mAP	Sim	Comparativo
A18	*Robust Deep Learning Methodto Detect Face Masks.* (Li, Cao and Zhang, 2020)	YOLOv3	WiderFace e Mafa	mAP	Não	Aplicação
A19	*Real-time Face Mask Detection in Video Data*. (Ding, Li and Yastremsky, 2021)	YOLOv5	Maskedface-Net e Flickr	mAP	Sim	Comparativo

A20	*Application of Yolo onMask Detection Task*. (Liu and Ren, 2021)	YOLOv3 e Faster R-CNN	Kaggle	*F1-score, Precision*	Não	Comparativo
A21	*Na Automatic System to Monitor the Physical Distance and Face MaskWearing of Construction Workers in COVID-19 Pandemic.* (Razavi et al., 2021)	Faster R-CNN e SSD	*Dataset* Customizado	mAP	Sim	Comparativo
A22	*WearMask: Fast In-browser Face Mask Detection with Serverless Edge Computing for COVID-19.* (Wang et al., 2021)	YOLOv1	Wider Face, Mafa, RMFD e MMD.	mAP	Sim	Aplicação
	Esta dissertação	Faster R-CNN, SSD e YOLO	Prajna	mAP	Sim	Comparativo

Tabela 5 – Artigos selecionados da revisão da literatura.

Baseado na Tabela 5, mostra a quantidade de trabalhos usando os métodos de detecção de Faster R-CNN, SSD e YOLO na de detecção de máscara. Entre os mais utilizados destaca-se o Faster R-CNN, SSD e YOLOv3, conforme Figura 10.

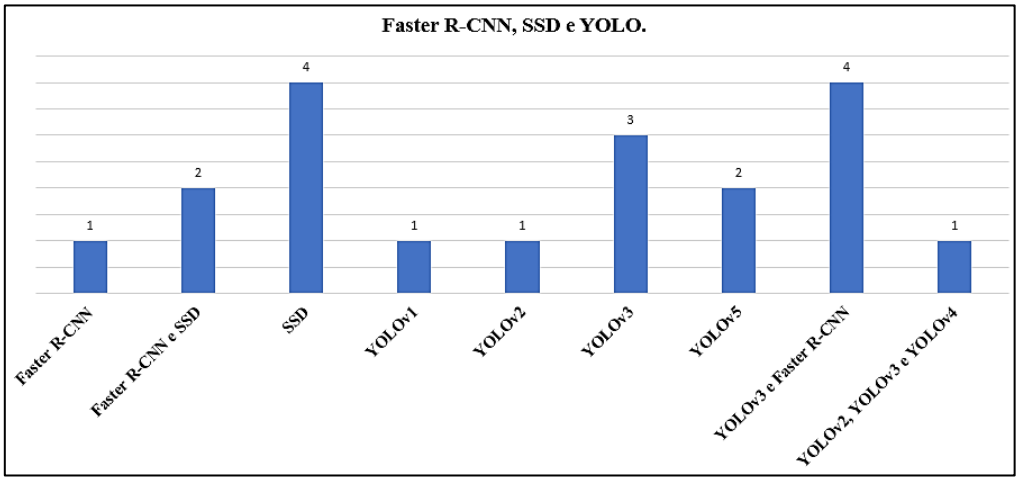


Figura 10 – Uso do Faster R-CNN, SSD e YOLO da revisão da literatura.

Na Tabela 5, o campo “método” indica o tipo de experimento adotado e como foi usado os algoritmos de detecção de objetos Faster R-CNN, SSD ou YOLO, se foi aplicado em um *dataset* ou se além disso foi comparado com outros detectores. Os estudos que aplicam os métodos de detecção em um *dataset* somam 27% (seis estudos) e que além disso ainda comparam com outro detector somam 59% (13 estudos). Estudos do tipo *survey* somam 14% (três estudos).

Na Tabela 5, o campo “*Transfer Learning*” indica se a proposta fez uso de transferência de aprendizagem. Os estudos que usam transferência de aprendizagem somam 59% (13 estudos).

Na Tabela 5, o campo “medida de avaliação” indica as métricas utilizadas para avaliar a taxa de acerto do modelo. Os estudos selecionados mostram 55% (12 estudos) dos trabalhos usando o mAP como métrica de avaliação, conforme Figura 11.

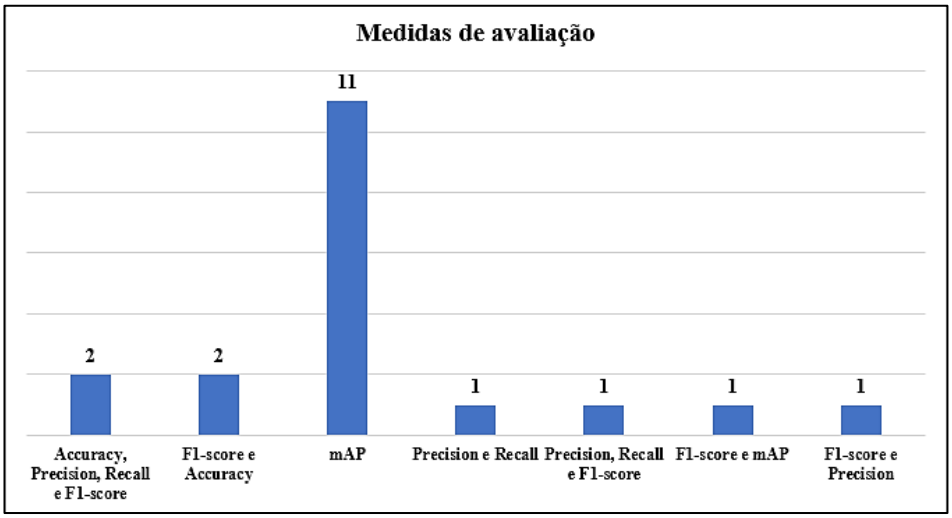


Figura 11 – Medidas de avaliação da taxa de acerto.

Baseado na análise dos estudos da Tabela 5, a Figura 12, a seguir, mostra os *datasets* identificados. Foram identificados *datasets* dedicados a detecção de máscara e de uso geral.

- Dedicados a detecção de máscara – Kaggle, Prajna, RMFD, MAFA, MaskedFace-Net, WMD e MMD;
- Uso geral – WIDER FACE e Flickr-Faces-HQ.

Os *datasets* customizados são formados por partes de dois ou mais *datasets* dedicados a detecção de máscara e aparecem em cinco trabalhos (A1, A7, A9, A17, A21).

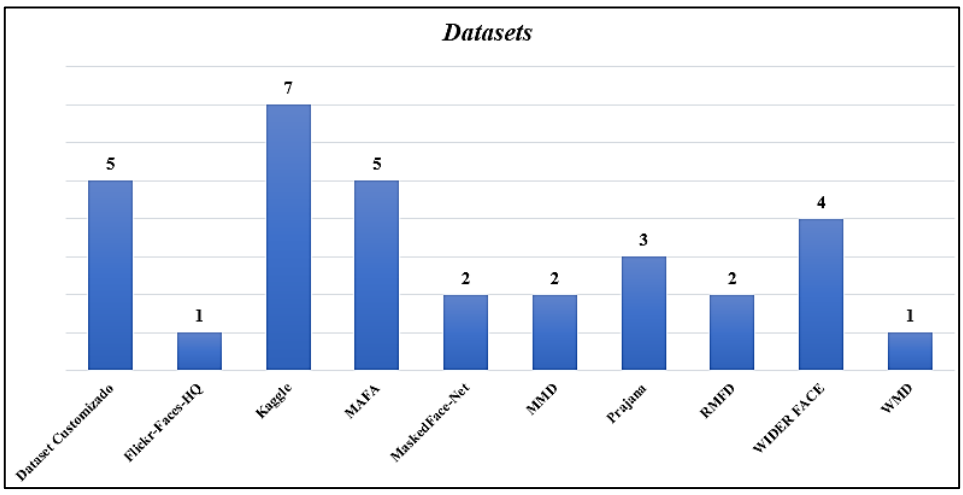


Figura 12 – *Dataset* identificados na revisão da literatura.

A Tabela 6, a seguir, mostra as características dos *datasets*, incluindo título e a

citação, o número de categorias, a distribuição de imagens das categorias e se a imagem contém uma ou mais de uma instância do objeto anotado na imagem.

ID	*Dataset*	#Classe	#Imagens	#instâncias por imagem
B1	Kaggle *Dataset*. (Kaggle, 2021)	3	853 imagens (com máscara, sem máscara e máscara vestidas incorretamente)	Vários
B2	Prajna *Dataset*. (Bhandary, 2020)	2	690 imagens com máscara 686 imagens sem mascara	Um
B3	RMFD - *Real-World Masked Face Dataset*. (X-zhangyang, 2022)	2	5.000 imagens com máscara 90.000 imagens sem mascara	Um e vários
B4	MAFA –*MaskedFAces*. (Ge et al., 2022)	2	35.806 imagens com máscara 30.811 imagens sem máscara	Um
B5	MaskedFace-Net. (Cabani et al., 2021)	3	67.049 vestida corretamente 66.734 vestida incorretamente	Um
B6	WMD- *WearingMask Detection*. (Wang, Zhao and Chen, 2021)	1	7.804 imagens com máscara	Vários
B7	MMD - *Medical Masks Dataset*. (Loey et al., 2021)	1	682 imagens com máscara	Vários
B8	Wider face. (Shuoyang1213, 2022)	1	32.203 de rosto de pessoas	Vários
B9	Flickr-Faces-HQ. (Flickr-Faces, 2022)	1	70.000 de rosto de pessoas	Uma

Tabela 6 – Caracteristicas dos *datasets* da revisão da literatura.

3.3 Discussão da Revisão

A principal métrica dos estudos é avaliar a taxa de acerto. Porém, dos trabalhos analisados, sete artigos (A6, A9, A14, A17, A18, A19 e A21) avaliam o desempenho dos detectores, também, em relação ao tempo de processamento de uma imagem nas detecções, indicado como tempo de inferência e nenhum artigo estudado aborda o tempo de treinamento como métrica para avaliar um detector.

A medida de avaliação em relação a taxa de acerto que se destacou foi o mAP. A justificativa é que a métrica lida melhor com o desequilíbrio dos casos verdadeiros e dos casos falsos, pois é esperado apenas detecções relevantes e que todos os objetos

anotados sejam detectados (Nasri et al., 2021).

Foi observado na análise dos artigos que oclusão é uma discussão recorrente nas propostas. Fazem menção a esse tipo de problema os artigos: A4, A7, A9, A14, A15, A18.

A revisão contribuiu para identificar o YOLOv4 como o estado da arte na tarefa de detecção de objetos, testado no *dataset* MS COCO. Com a revisão foi possível ter conhecimento do uso dos principais algoritmos de detecção de objetos na tarefa de detecção de máscara Faster R-CNN, SSD e YOLO, dos *datasets* existentes na tarefa de detecção de máscara e de uso geral, bem como a medida de avaliação mais utilizada o mAP.

3.4 Trabalhos Relacionados

Os trabalhos selecionados da Tabela 5 “Artigos selecionados da revisão da literatura” foram reescritos em uma nova tabela com os principais resultados dos algoritmos de detecção estudados nesta dissertação. Primeiramente foi removido os artigos de revisão (*survey*), depois foi apresentado a nova tabela.

A Tabela 7, a seguir, mostra os principais resultados dos artigos analisados. A coluna “ID” representa o título e o autor apresentados anteriormente na seção 3.2. A segunda coluna “#imagens” é a quantidade de imagens que foi utilizada nos *datasets* apresentados nos artigos. A terceira coluna representa a divisão treino e teste, a quarta coluna o valor do *batch size*. A quinta coluna “Iterações / Épocas”, que representa por quanto tempo o algoritmo foi treinado, alguns trabalhos fornecem o valor em iterações e outros em épocas. A sexta coluna representa o algoritmo de detecção utilizado com base na proposta do artigo e por último, a coluna que apresenta o resultado com a respectiva medida de avaliação.

ID	**#Imagens**	**Divisão treino e teste**	***Batch Size***	**Iterações / Épocas**	**Algoritmo de detecção**	**Medida de avaliação**				
						mAP	**Accuracy**	**F1-Score**	**Precision**	**Recall**
A1	8.805	80/20	6	Iterações 120k e 150k	YOLOv3	92%				
					Faster R-CNN	86,9%				
A2	2.229	80/20	não informado	100 Épocas	SSDMNV2		92,64 %	93%		
A3	1.376	não informado	32	20 Épocas	SSD		99%	99%		
A4	1.535	70/30	6	60 Épocas	YOLOv2	81%				
A5	5.211	80/20	32	20 Épocas	SSD		100%	100%	100%	99%
A6	7.500	não informado	não informado	70 Épocas	YOLOv3	55%				
					Faster R-CNN	62%				
A7	52.635	80/20	64	Iterações 8k	YOLOv4	64,31%				
A9	11.792	90/10	não informado		YOLOv3		93,2%	95,1%	94,6%	95,7%
A10	853	70/30	64	400 Épocas	YOLOv5	81%				
A11	7.000	5000 treino e 2000 teste	8	75 Épocas	YOLOv3				83 %	90,1%

A12	1.594	não informado	não informado	Iterações 200k	Faster R-CNN			96,47%	94,62%	98,40%
					SSD			94,20%,	97,64%	91%
A14	4.500	4000 treino e 500 teste	4	120 Épocas	SSD-Mask	94,1%				
A16	1.784	80/20	não informado	Iterações 200k	Faster R-CNN	91,4%		91,1%		
A17	2.500	*Cross Validation* (*k = 3*)	não informado	não informado	YOLOv3	26,8%				
					Faster R-CNN	28,1%				
A18	9.886	70/30	não informado	não informado	YOLOv3	86,3%				
A19	180.000	80/20	não informado	não informado	YOLOv5	89,8%				
A20	853	Proporção treino e teste 4:1	64	200 Épocas	YOLOv3			70,6%	90,5%	
					Faster R-CNN			72,1%	93,2%	
A21	3.300	80/20	1	Iterações 200k	Faster R-CNN	99,8%				
					SSD	82%				
A22	9.097	80/20	16	120 Épocas	YOLO	89%				

Tabela 7 – Resultados dos trabalhos relacionados.

A proposta dos artigos A2, A3, A5 e A9 fazem uso da arquitetura CNN Mobilenet v2, sendo a CNN mais utilizada dos artigos estudados.

No estudo A2, a proposta é apresentada como SSDMNV2, onde é utilizado método de detecção SSD com uma CNN Resnet-10 para a detecção do rosto e posteriormente um Mobilenet-v2 faz a classificação com ou sem máscara.

O artigo A14 apresenta uma proposta chamado SSD-Mask, desenvolvido a partir do método de detecção de objetos SSD. Quando comparado com o SSD original (baseado na CNN VGG-16), o SSD-Mask supera em taxa de acerto e velocidade.

O estudo A21 utiliza a mesma metodologia deste trabalho. O artigo compara cinco modelos pré-treinados da *API Tensorflow Object Detection*. Foram três CNNs com o Faster R-CNN e duas com o SSD. O Faster R-CNN se destacou com a CNN Inception-Resnet-v2, enquanto o SSD se destacou com o ResNet50 v1 FPN.

Portanto, os trabalhos relacionados mostraram os principais resultados na eficiência dos métodos de detecção de máscara baseado em redes neurais usando o

Faster R-CNN, SSD e YOLO.

Considerações finais

Neste capítulo foi apresentado a revisão da literatura e os principais resultados com os trabalhos relacionados. A revisão possibilitou identificar os principais algoritmos de detecção de objetos Faster R-CNN, SSD e YOLO e como eles foram utilizados em trabalhos detecção de máscara. avaliados. Além disso, mostram excelentes resultados nesse tipo de tarefa em diferentes métricas.

Este trabalho se diferencia dos trabalhos relacionados com um método comparativo entre nove modelos pré-treinados de arquiteturas de detecção e adiciona o tempo de treinamento na avaliação do detector de máscara mais eficiente.

Capítulo 4 – Descrição e Preparação do Ambiente de Testes

Neste capítulo serão apresentados os recursos e a preparação do ambiente para os experimentos. Na seção 4.1 o banco de dados, 4.2 a lista de modelos pré-treinados, 4.3 definições dos hiper parâmetros e 4.4 a definição dos métodos de avaliação.

Os frameworks utilizados neste trabalho são: Tensorflow *API Object Detection* e o *Darknet*. Ambos os frameworks têm uma comunidade muito ativa e diversos materiais que dão sustentação às ferramentas, isso motivou a escolha e o emprego deles no presente trabalho. Além disso, foi utilizado o ambiente do Google Colaboratory na versão PRO, pois ele traz a vantagem da linguagem Python, várias dependências instaladas e mais tempo de acesso a GPU.

A descrição detalhada do ambiente, organização dos diretórios, os ajustes nos arquivos de configuração, scripts e os pesos das redes neurais estão disponíveis no Github deste trabalho em https://github.com/mjf2004/projeto-mestrado. A Tabela 8 mostra de forma resumida as etapas do fluxo de trabalho, a coluna descrição indica a tarefa realizada.

ID	Descrição	
1	Banco de dados	Organizar as imagens do *dataset* em dois diretórios representando os objetos com máscara e sem máscara;
2		Dividir o *dataset* em conjuntos de treino e teste;
3		Fazer anotação da instância do objeto em cada imagem;
4	*Frameworks* e APIs	Criar os arquivos auxiliares de configuração dos *frameworks*;
5		Downloads dos pesos dos modelos pré-treinados;
6		Instalação dos *frameworks* e APIs;
7		Configuração dos hiper parâmetros de treinamento no arquivo de config de cada *frameworks*;
8	Treinamento e teste	Executar o treinamento e avaliação com os dados de teste;
9		Exportar o modelo para fazer as inferências.

Tabela 8 - Fluxo de trabalho.

4.1 Banco de dados

Para os experimentos foram utilizadas imagens de rostos de pessoas com

máscara e sem máscara. A região do rosto entre o nariz, boca e o queixo devem estar totalmente cobertos para que seja caracterizado um rosto com máscara.

Os experimentos foram realizados com o conjunto de imagens Prajna *Dataset*. Esse conjunto de imagens foi um dos primeiros *datasets* disponibilizados para a comunidade na detecção de máscaras e foi adotado neste projeto deste seu início. Esse *dataset* tem apenas uma instância do objeto por imagem, isso facilitou a rotulação dos objetos a serem detectados. Os dados de imagens está em https://github.com/prajnasb/observations (Bhandary, 2020). No diretório "*experiments*", o subdiretório "*data*" contém as imagens nos subdiretórios "*with_mask*" e "*without_mask*". As imagens estão armazenadas no formato JPG e em RGB de intensidade de 24 bits.

Foram eliminadas quatro imagens do subdiretório "*with_mask*". Assim, o número de imagens está disposto da seguinte maneira: 686 imagens com máscara e 686 imagens sem máscara e em ambos os casos contém imagens resultantes das técnicas de aumento de dados como rotação, brilho, inversão etc. Os exemplos podem ser vistos no apêndices 1 e 2 deste trabalho.

Na fase de pré-processamento as imagens de entrada são redimensionadas para um tamanho fixo. Para auxiliar na escolha do tamanho da imagem de entrada dos modelos pré-treinados, foi investigada a largura e altura das imagens do *dataset*. O objetivo é evitar perdas desse redimensionamento quando a imagem for enviada para a fase de extração de características. Como pode ser observado na Tabela 9, a maior parte das imagens são de 300 pixels a 600 pixels.

	Com máscara		Sem máscara	
	Altura	Largura	Altura	Largura
100 a 299 pixels	32,5%	37%	31%	35%
300 a 599 pixels	65%	57%	63%	60%
>= 600 pixels	2,5%	6%	6%	5%

Tabela 9 – Distribuição da altura e largura das imagens.

Para o treino e teste, foi utilizado a estratégia 80-20 para dividir o *dataset*, ou seja, a base de imagens com máscara e da base de imagens sem máscara foi dividido em 80% de imagens para treinamento e 20% para teste, sendo respectivamente 1098 imagens de treinamento e 274 imagens de teste (utilizado para gerar a avaliação com a métrica mAP).

Anotação das imagens

Para atender ao requisito para o treinamento da rede neural, se faz necessária a etapa de pré-processamento de anotar a região de interesse na imagem e rotular as categorias de acordo com o objeto a ser detectado, as duas categorias são: “com_mascara” e “sem_mascara”. Essa tarefa foi realizada manualmente utilizando a ferramenta LabelImg versão 1.8.2.

Existem padrões de anotações já pré-estabelecidas como: o formato Pascal VOC em .XML, COCO em .JSON e o YOLO em .TXT (Padilla et al., 2021). O padrão XML foi adotado pelo *framework Tensorflow API Object Detection* de acordo com o tutorial da própria ferramenta (TensorFlow 2 Object Detection API, 2021).

4.2 Modelos Pré-treinados

A Tabela 10 apresenta os modelos pré-treinados. A primeira coluna o framework utilizado, a segunda coluna o algoritmo de detecção, a terceira coluna a arquitetura CNN e a última o tamanho de imagem de entrada redimensionada no pré-processamento.

Framework	**Método de Detecção**	**Arquitetura CNN**	**Entrada**
Tensorflow API Object Detection	Faster R-CNN	ResNet50 v1	640×640
		ResNet101 v1	640×640
		Inception ResNet v2	640×640
	SSD	MobileNet v2	320×320
		MobileNet v2 FPNLite	320×320
		ResNet50 v1 FPN (RetinaNet)	640×640
Darknet	Yolov2	Darknet-19	416×416
	Yolov3	Darknet-53	416×416
	Yolov4	CSP-Darknet-53	608×608

Tabela 10: Modelos pré -treinados.

4.3 Definição dos Hiper parâmetros

Os hiper parâmetros de treinamento ajustados para este projeto são basicamente o tamanho do lote (*batch size*) e a quantidade de iterações, definidos da seguinte forma:

- O tamanho do lote adotado para o treinamento foi o máximo permitido pelo ambiente Google Colaboratory, pois para alguns modelos pré-treinados na tentativa de aumentar o *batch size* ocorria erro de estouro de memória.
- Os estudos não apresentam uma resposta conclusiva sobre o número de iterações ideal. Portanto, foi definido o número fixo de iterações em cada arquitetura conforme testes preliminares, pois os *frameworks* utilizados não trazem implementados técnicas de parada antecipada, quando a rede começa a sofrer *overfitting*, como é o caso da técnica *Early Stopping*, por exemplo (Bengio, 2016).

É comum na literatura a presença do número de épocas de treinamento para indicar por quanto tempo a rede será treinada, mas os *framework*s utilizados neste trabalho permitem configurar apenas o número de iterações. No contexto deste trabalho, uma época é concluída quando a rede neural passou por todas as imagens de treinamento. A cada iteração é carregado um lote de imagens, definido pelo hiper parâmetro *batch size*. Logo, o número de imagens de treinamento dividido pelo *batch size* resulta na quantidade de iterações para uma época (Androidkt, 2019).

4.4 Definição dos Métodos de Avaliação

Os métodos de avaliação do detector de máscara deste projeto é a taxa de acerto, o tempo de total de treinamento e o tempo de inferência. Essas métricas são prioridades para a definir o detector de máscara mais eficiente deste projeto. As considerações acerca das métricas utilizadas são descritas a seguir.

Taxa de acerto

Para avaliar a taxa de acerto foi escolhido a métrica das competições em detecção de objetos *Pascal VOC*, que desde 2010 passou a utilizar a interpolação de todos os pontos para calcular mAP@50 (*e.g.* IOU 50%) por categoria (Everingham et al., 2015). A métrica *Pascal VOC* com a *API Tensorflow Object Detection* é passível de configuração no arquivo de configuração do modelo pré-treinado. O YOLO já tem essa configuração padrão. Além de ser a métrica mais usada em detectores de objetos, ela é a

métrica em comum presente nos dois *frameworks* usados no trabalho.

Foi identificado nos experimentos iniciais que a implementação da *API Tensorflow Object Detection* tem falha de desenvolvimento na métrica Pascal VOC ao mostrar baixo mAP em uma das categorias. A correção se encontra no repositório oficial da *API*, na seção de contribuições da comunidade e o arquivo foi substituído para o processo de teste e avaliação do modelo (TensorFlow, 2021).

Tempo de treinamento

Para as arquiteturas com o Faster R-CNN e o SSD, o tempo de treinamento pôde ser retirado através da ferramenta de geração de gráfico *Tensorboard*, cujo recurso é disponibilizado pela própria *Tensorflow API Object Detection*. No YOLO, foi calculado o tempo de treinamento a partir do total de iterações e o tempo médio de uma iteração.

Inferência

Neste trabalho, cinco imagens foram selecionadas para verificar a qualidade das detecções. As inferências foram realizadas em diferentes critérios, são eles: “oclusão”, “generalização com máscara” e “generalização sem máscara”, “imagem do *dataset* com máscara” e “imagem do *dataset* sem máscara”. Em cada critério o detector deve mostrar a categoria correta com máscara ou sem máscara.

Conforme Figura 13, as imagens “oclusão”, “generalização com máscara” e “generalização sem máscara” (do autor) são as imagens que estão fora do Prajna *dataset*. Cada imagem está nomeada de acordo com o seu critério de avaliação.

Figura 13 - Imagens utilizadas para inferência.

A resolução de cada imagem é mostrada na Tabela 11.

ID	Resolução	Critério avaliado
1	705px × 778px	Oclusão
2	697px × 746px	Generalização com máscara
3	695px × 846px	Generalização sem máscara
4	285px × 372px	*dataset* com Máscara
5	318px × 442px	*dataset* sem Máscara

Tabela 11 – Resolução e critério de inferência das imagens.

Foi utilizado o tempo médio das inferências das cinco imagens e calculado o valor de FPS aproximado (considerando a definição 1000ms = 1s). O limite IOU de 0.5 foi utilizado para todos os testes. Em cada imagem foi verificado a quantidade de verdadeiro positivo, falso positivo e falso negativo para a seção de análise e discussão.

No *script* utilizado para fazer a inferência dos modelos gerados a partir da *API Object Detection* foram adicionados três linhas de código extras. A primeira linha é o método python TIME(), que pega a data e hora atual e é inserida após o carregamento do modelo para marcar o início da contagem de tempo de processar a imagem. A segunda linha é o mesmo método inserido após a finalização da inferência e a terceira linha é a diferença entre as duas para armazenar o tempo de inferência. As linhas estão destacadas na cor amarelo no Apêndice 3 deste trabalho.

Erro (custo)

O *Tensorboard* e o gráfico do YOLO fornecem informações importantes como o erro (custo). O erro indica o quanto a rede neural está distante da resposta correta e o quanto deve ser ajustado os pesos através do algoritmo de *backpropagation*. O valor do erro foi apresentado em cada experimento para indicar o desempenho da fase de treinamento e o quanto se aproximou da função objetivo.

Considerações finais

No presente capítulo foram apresentados os componentes que fazem parte do referencial metodológico deste trabalho. Foram mostradas as etapas que vão permitir ao final do projeto escolher um detector de máscara eficiente em termos de taxa de acerto, tempo de treinamento e tempo de inferência. Para isso, foi apresentado o fluxo de trabalho, o *dataset* e a definição dos métodos de avaliação para avaliar o melhor detector, dentro das condições deste trabalho.

Capítulo 5 – Experimentos e Discussão dos Resultados

Este capítulo apresenta o uso dos algoritmos de detecção de objetos e as respectivas CNNs no contexto de detecção de máscara.

Inicialmente temos na seção 5.1 as definições gerais dos experimentos. Na seção 5.2 são apresentados os hiper parâmetros retirados diretamente dos arquivos de configuração dos *frameworks* (nenhuma alteração realizada). Em seguida, os experimentos na seção 5.3. A síntese dos experimentos é mostrado na seção 5.4 e, nas seções 5.5 e 5.6 a discussão dos resultados e os resultados comparados com os encontrados na literatura, respectivamente.

5.1 Definições gerais dos experimentos

O hardware considerado para todos os experimentos, tanto para o treinamento e avaliação, quanto para a inferência são as seguintes: 2 Processadores Intel(R) Xeon(R) CPU @ 2.20GHz, 25 GB de memória RAM e placa de vídeo Tesla P100-PCIE; informações retiradas no terminal do Google Colaboratory.

Em todos os experimentos, o desempenho em relação ao custo é apresentado. Além disso, será mostrada a taxa de acerto com a métrica mAP@50 e a qualidade das detecções em cinco imagens, simulando testes de produção. No Faster R-CNN e SSD as caixas em verde representam a categoria “com_mascara” e as caixas em amarelo a categoria “sem_mascara”. No YOLO, as caixas em roxo representam a categoria “com_mascara” e as caixas em verde a categoria “sem_mascara”.

5.2 Hiper Parâmetros do Otimizador e Âncoras

Os experimentados foram realizados utilizando os valores padrão dos arquivos de configuração dos modelos pré-treinados, apresentados nas tabelas 12, 13 e 14.

	Otimizador SGD		**Âncoras**		**Limiar NMS**
Detector	**Cosine Decay Learning Rate**	**Momentum**	**Escalas**	**Proporções**	
Faster R-CNN Resnet50 v1	Começa em 0.013 e termina em 0.04 após 2k iterações.	0.9	0.25, 0.5, 1 e 2.	0.5, 1, 2	RPN = 0,7 NMS = 0,6
Faster R-CNN Resnet101 v1	Começa em 0.013 e termina em 0.04 após 2k iterações.	0.9	0.25, 0.5, 1 e 2.	0.5, 1, 2	RPN = 0,7 NMS = 0,6
Faster R-CNN Inception Resnet v2	Começa em 0 e termina em 0.16 após 2.5k iterações.	0.9	0.25, 0.5, 1 e 2.	0.5, 1, 2	RPN = 0,7 NMS = 0,6

Tabela 12 - Conjunto de hiper parâmetros Faster R-CNN.

	Otimizador SGD		**Âncoras**		**Limiar NMS**
Detector	**Cosine Decay Learning Rate**	**Momentum**	**Escalas**	**Proporções**	
SSD Mobilenet v2	Começa em 0.13 e termina em 0.8 após 2k iterações.	0.9	Mínimo: 0,2 Máxima: 0,95	1.0, 2.0, 0.5, 3.0 e 0.33	0,6
SSD Mobilenet v2 FPNLite	Começa em 0.03 e termina em 0.08 após 2k iterações.	0.9	Multi Escala FPN $4 \times (2^3, \ldots, 2^7)$	0.5, 1, 2	0,6
SSD Resnet 50 v1 FPN (RetinaNet)	Começa em 0.013 e termina em 0.04 após 2k iterações.	0.9	Multi Escala FPN $4 \times (2^3, \ldots, 2^7)$	0.5, 1, 2	0,6

Tabela 13 - Conjunto de hiper parâmetros SSD.

	Otimizador SGD			**#Filtros (#classes + 5) × #âncoras**	**Limiar NMS**
Detector	**Learning Rate**	**Momentum**	**Weight Decay**		
YOLOv2	0.001	0.9	0.0005	(2 + 5) × 5 = 35	0,6
YOLOv3	0.001	0.9	0.0005	(2 + 5) × 3 = 21	0,6
YOLOv4	0.0013	0.949	0.0005	(2 + 5) × 3 = 21	0,7

Tabela 14 - Conjunto de hiper parâmetros YOLO.

5.3 Experimentos

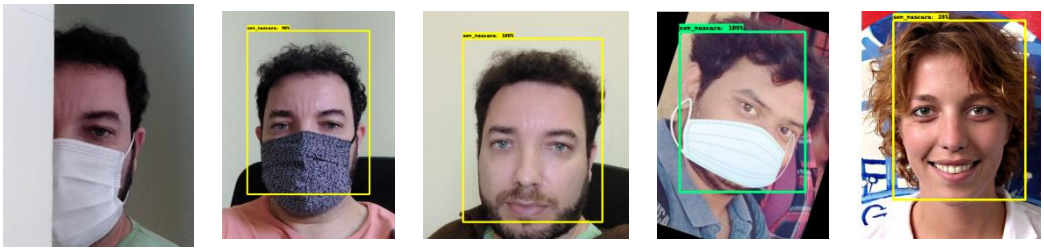
Nos experimentos 1 a 9 foi usado uma base 80-20, representando 80% das imagens para treino e 20% das imagens para teste (treinados com 1.098 do total de 1.372 imagens). A Tabela 15 mostra os hiper parâmetros de treinamento. Os hiper parâmetros de treinamentos são: *batch size*, número de iterações e épocas.

Experimento	Método + CNN	*Batch size*	#Iterações	#Épocas
1	Faster R-CNN Resnet50 v1	1	96k	88
2	Faster R-CNN Resnet101 v1	1	96k	88
3	Faster R-CNN Inception-Resnet-v2	1	96k	88
4	SSD Mobilenet v2	8	60k	435
5	SSD Mobilenet v2 FPNLite	64	12k	667
6	SSD Resnet 50 v1 FPN	8	60k	435
7	YOLOv2	64	1.5k	84
8	YOLOv3	64	1.5k	84
9	YOLOv4	64	1.5k	84

Tabela 15 – Hiper parâmetros para a base 80-20.

Experimento 1: Faster R-CNN Resnet50 v1

Como resultado do experimento 1, foi apresentado um AP@50 da classe com máscara de 98,47% e da classe sem máscara foi de 98,45%, resultando em um mAP@50 de 98,46%. A rede apresentou um erro de 0,0380 no final do treinamento. Como pode ser visto na Figura 14, a imagem com “oclusão” gerou um falso negativo e a imagem “generalização com máscara” gerou um falso positivo.

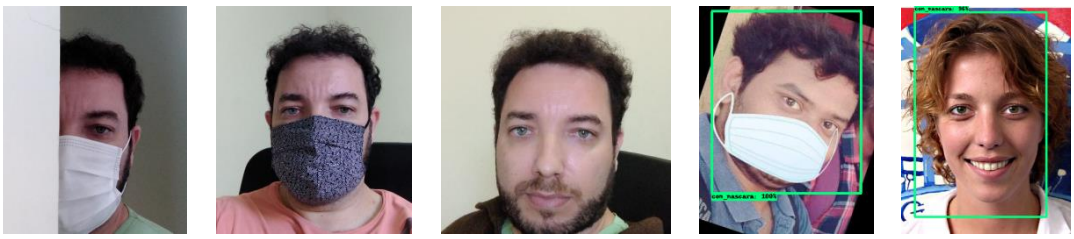


Oclusão | *Generalização Com Máscara* | *Generalização Sem máscara* | *Do dataset com máscara* | *Do dataset sem máscara*

Figura 14 – Inferência Faster R-CNN Resnet50 v1 base 80-20.

Experimento 2: Faster R-CNN Resnet101 v1

Como resultado do experimento 2, foi apresentado um AP@50 da classe com máscara de 97,69% e da classe sem máscara foi de 96,45%, resultando em um mAP@50 de 97,07%. A rede apresentou um erro de 4,6341e^{-3} no final do treinamento. Conforme Figura 15, as imagens "oclusão", "generalização com máscara", "generalização sem máscara" geraram um falso negativo em cada um deles. A imagem "sem máscara do *dataset*" gerou um falso negativo e um falso positivo.

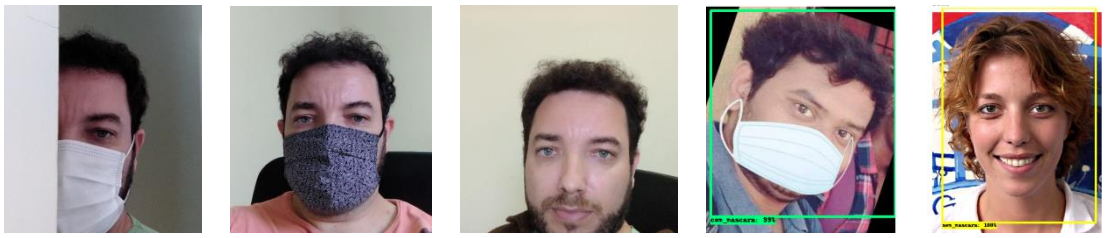


Oclusão | *Generalização Com Máscara* | *Generalização Sem máscara* | *Do dataset com máscara* | *Do dataset sem máscara*

Figura 15 – Inferência Faster R-CNN Resnet101 v1 base 80-20.

Experimento 3: Faster R-CNN Inception-Resnet-v2

Como resultado do experimento 3, foi apresentado um AP@50 da classe com máscara de 93% e da classe sem máscara foi de 85,89%, resultando em um mAP@50 de 89,44%. A rede apresentou um erro de 1,013 no final do treinamento. Conforme Figura 16, imagens "oclusão", "generalização com máscara", "generalização sem máscara" geraram um falso negativo em cada um deles.

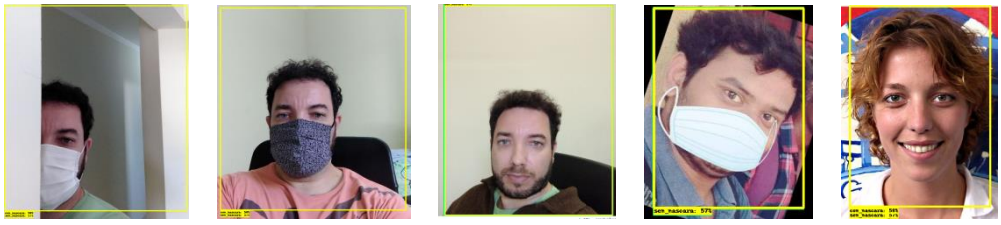


Oclusão | *Generalização Com Máscara* | *Generalização Sem máscara* | *Do dataset com máscara* | *Do dataset sem máscara*

Figura 16 – Inferência Faster R-CNN Inception-Resnet-v2 base 80-20.

Experimento 4: SSD Mobilenet v2

Como resultado do experimento 4, foi apresentado um AP@50 da classe com máscara de 47,94% e da classe sem máscara foi de 33,45%, resultando em um mAP@50 de 40,69%. A rede apresentou um erro de 7,4021e^{+4} no final do treinamento. Conforme visto na Figura 17, as imagens "oclusão" e "generalização com máscara" geraram dois falsos positivos e um falso negativo em cada um deles. As imagens "generalização sem máscara", "*dataset* com máscara" e "*dataset* sem máscara" geraram um falso positivo cada e o verdadeiro positivo detectado em cada imagem ficou mal ajustado com o objeto.

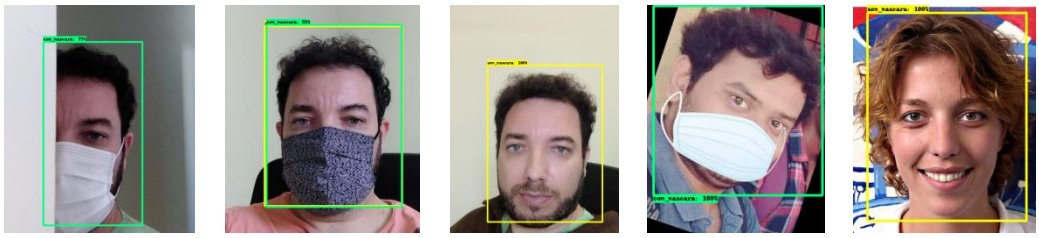


Oclusão | *Generalização Com Máscara* | *Generalização Sem máscara* | *Do dataset com máscara* | *Do dataset sem máscara*

Figura 17 – Inferência SSD Mobilenet v2 base 80-20.

Experimento 5: SSD Mobilenet v2 FPNLite

Como resultado do experimento 5, foi apresentado um AP@50 da classe com máscara de 100% e da classe sem máscara foi de 100%, resultando em um mAP@50 de 100%. A rede apresentou um erro de 0,1305 no final do treinamento. A qualidade das detecções pode ser observada na Figura 18, onde apenas a imagem "generalização com máscara" gerou um falso positivo.

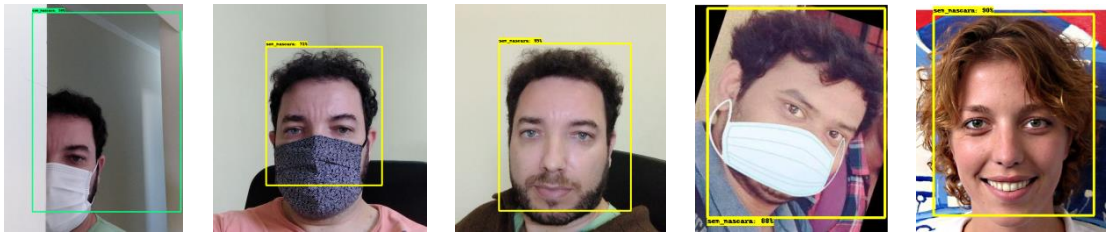


Oclusão | *Generalização Com Máscara* | *Generalização Sem máscara* | *Do dataset com máscara* | *Do dataset sem máscara*

Figura 18 – Inferência SSD Mobilenet v2 FPNLite base 80-20.

Experimento 6: SSD Resnet50 v1 FPN

Como resultado do experimento 6, foi apresentado um AP@50 da classe com máscara de 99,97% e da classe sem máscara foi de 98,45%, resultando em um mAP@50 de 99,92%. A rede apresentou um erro de 0,2715 no final do treinamento. A qualidade das detecções pode ser observada na Figura 19, onde a imagem com "oclusão" embora tenha gerado um verdadeiro positivo a caixa ficou mal ajustada. As imagens "generalização com máscara" e "*dataset* com máscara" geraram um falso positivo e um falso negativo em cada um deles.

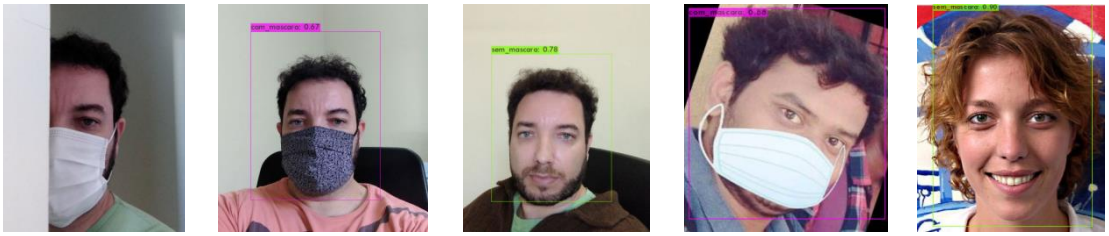


Oclusão | *Generalização Com Máscara* | *Generalização Sem máscara* | *Do dataset com máscara* | *Do dataset sem máscara*

Figura 19 – Inferência SSD Resnet50 v1 FPN base 80-20.

Experimento 7: YOLOv2

Como resultado do experimento 7, foi apresentado um AP@50 da classe com máscara de 99,85% e da classe sem máscara foi de 99,34%, resultando em um mAP@50 de 99,60%. A rede apresentou um erro de 0,0558 no final do treinamento. A qualidade das detecções pode ser observada na Figura 20, onde apenas a imagem com "oclusão" gerou um falso negativo.

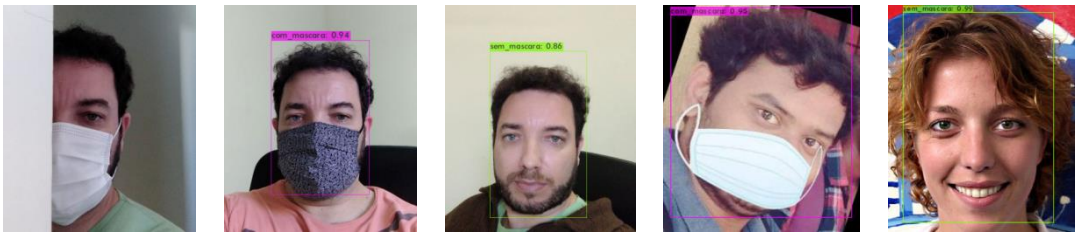


Oclusão | *Generalização Com Máscara* | *Generalização Sem máscara* | *Do dataset com máscara* | *Do dataset sem máscara*

Figura 20 – Inferência YOLOv2 base 80-20.

Experimento 8: YOLOv3

Como resultado do experimento 8, foi apresentado um AP@50 da classe com máscara de 92,39% e da classe sem máscara foi de 99,96%, resultando em um mAP@50 de 96,18%. A rede apresentou um erro de 0,0833 no final do treinamento. A qualidade das detecções pode ser observada na Figura 21, onde apenas a imagem com "oclusão" gerou um falso negativo.

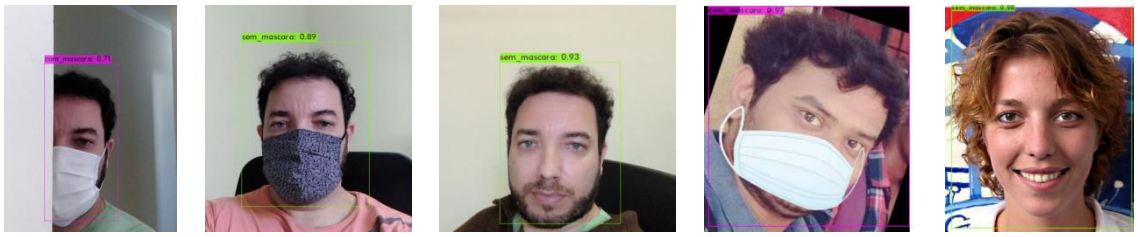


Oclusão | *Generalização Com Máscara* | *Generalização Sem máscara* | *Do dataset com máscara* | *Do dataset sem máscara*

Figura 21 – Inferência YOLOv3 base 80-20.

Experimento 9: YOLOv4

Como resultado do experimento 9, foi apresentado um AP@50 da classe com máscara de 100% e da classe sem máscara foi de 99,89%, resultando em um mAP@50 de 99,95%. A rede apresentou um erro de 0,7649 no final do treinamento. A qualidade das detecções pode ser observada na Figura 22, onde todas as imagens detectaram os verdadeiro positivos corretamente.



Oclusão | *Generalização Com máscara* | *Generalização Sem Máscara* | *Do dataset com máscara* | *Do dataset sem máscara*

Figura 22 – Inferência YOLOv4 base 80-20.

5.4 Síntese dos Experimentos

A Tabela 16 mostra a síntese dos experimentos realizados. A primeira coluna (Método + CNN) representa o método de detecção e a respectiva arquitetura CNN. A segunda coluna, o hiper parâmetro *batch size*. A terceira coluna refere-se ao número de épocas e a quarta coluna mostra a avaliação usando a métrica mAP@50, onde @50 significa 50% de IOU para uma detecção correta como verdadeiro positivo.

Método + CNN	***Batch Size***	**#Épocas**	**mAP@50**
Faster R-CNN Resnet 50	1	88	98,46%
Faster R-CNN Resnet 101	1	88	97,07%
Faster R-CNN Inception-Resnet-v2	1	88	89,44%
SSD Mobilenet v2	8	435	40,69%
SSD Mobilenet v2 FPNLite	**64**	**667**	**100%**
SSD Resnet 50 v1 FPN	8	435	99,92%
YOLOv2	64	89	99,60%
YOLOv3	64	89	96,18%
YOLOv4	**64**	**89**	**99,95%**

Tabela 16 – Síntese dos experimentos realizados com a base 80-20.

O mAP em praticamente todas as arquiteturas demonstrou resultados excelentes na taxa de acerto, com mAP@50 acima dos 96%. A exceção é notada no SSD Mobilenet v2 com mAP@50 de 40,9%. Os melhores resultados e inferências foram com os modelos SSD MobileNet v2 FPNLite e o YOLOv4, inclusive foram os únicos a detectarem oclusão.

O poder de detecção é o elemento essencial para analisar os outros critérios de avaliação do detector, pois de nada adianta processar rápido uma imagem ou treinar num tempo satisfatório, se a qualidade da detecção é ruim.

A Tabela 17 mostra o tempo médio de cada iteração, tempo total de treinamento e o tempo de processar uma imagem.

	Treinamento			Inferência	
Método + CNN	**#Iterações**	**Tempo médio (por Iteração)**	**Tempo total de treinamento**	**Tempo médio (por Inferência)**	**FPS**
Faster R-CNN Resnet 50	96k	0,109s	2h55m57s	0,110s	9
Faster R-CNN Resnet 101	96k	0,150s	4h00m49s	0,125s	8
Faster Inception-Resnet-v2	96k	0,417s	11h07m29s	0,378s	2
SSD Mobilenet v2	60k	0,093s	1h33m27s	0,060s	16
SSD Mobilenet v2 FPNLite	**12k**	**1,027s**	**3h25m25s**	**0,071s**	**14**
SSD Resnet-50 v1 FPN	60k	0,546s	9h06m12s	0,084s	11
YOLOv2	1,5k	4,215s	1h45m22s	0,008s	125
YOLOv3	1,5k	10,707s	4h27m40s	0,017s	58
YOLOv4	**1,5k**	**24,643s**	**10h16m04s**	**0,032s**	**31**

Tabela 17 – Desempenho de treinamento e inferência base 80-20.

Conforme visto na Tabela 17, o YOLOv4 processou a 31 FPS e o SSD Mobilenet v2 FPNLite as 14 FPS. Embora, ambas as arquiteturas detectem em tempo real, o YOLO é notoriamente mais rápido.

5.5 Análise e Discussão dos Resultados

Nesta seção foi realizado uma discussão e análise dos resultados obtidos dos experimentos. As discussões têm base empírica nas observações ao longo dos experimentos e considerações encontrada na literatura.

Discussão sobre a taxa de acerto

No geral, as arquiteturas tiveram excelentes resultados quando avaliados com a métrica mAP@50. Isso significa que as arquiteturas estão detectando bem, se submetidas às imagens ao qual as características estão estritamente presentes no *dataset*, pois os 20% da base de teste tem os mesmos padrões das imagens usadas no treinamento.

Segundo Shen (2018), o comportamento do treinamento varia de acordo com o conjunto de dados e os hiper parâmetros de treinamento e, ainda relata que não existe uma orientação geral para a escolha dos hiper parâmetros. Outro ponto notado é que a rede pode tomar diferentes caminhos na superfície de erro, isso porque as imagens dos lotes são enviadas de forma aleatória para o cálculo do erro. Logo, com um número fixo de iterações pode-se ter diferentes resultados com a mesma arquitetura.

Vários testes preliminares foram realizados. Em relação aos hiper parâmetros

batch size e total de iterações, o SSD Mobilenet v2 FPNLite e YOLOv4 foram as arquiteturas que deram os melhores resultados em todos os casos. As arquiteturas Faster R-CNN Inception-Resnet-v2 e Mobilenet v2 foram as que tiveram os piores resultados.

No experimento 4, o treinamento com a arquitetura Mobilenet v2 (sem FPN) mostrou um erro elevado de $7{,}4021e^{+4}$, que refletiu no péssimo desempenho nas detecções.

Discussão sobre a qualidade das inferências

Para verificar a velocidade e qualidade das detecções em caráter de produção foram usadas as cinco imagens mostradas em cada um dos experimentos. A Tabela 18 mostra a coluna VP, que representa a soma de verdadeiro positivos das cinco imagens. A coluna FP e FN representa a soma de falsos positivos e falsos negativos das cinco imagens, respectivamente.

Método + CNN	**VP**	**FP**	**FN**
Faster R-CNN Resnet 50	3	1	1
Faster R-CNN Resnet 101	1	1	4
Faster R-CNN Inception-Resnet-v2	2	0	3
SSD Mobilenet v2	3	7	2
SSD Mobilenet v2 FPNLite	**5**	**1**	**0**
SSD Resnet 50 v1 FPN	3	2	2
YOLOv2	4	0	1
YOLOv3	4	0	1
YOLOv4	**5**	**0**	**0**

Tabela 18 - Falsos positivos e falsos negativos.

Idealmente, espera-se que exista apenas um verdadeiro positivo em cada imagem, zero falso positivo e zero falso negativo. A presença de muitos falsos positivos e falsos negativos representa baixa qualidade do detector. Dessa forma, depois do modelo exportado, as melhores inferências foram com os modelos SSD Mobilenet v2 FPNLite e o YOLOv4.

Falsos positivos: geralmente, estão relacionados com a falta de variedade no conjunto de dados de treinamento. Segundo CHENG (2019), a CNN precisa treinar o objeto em diversos contextos e deve aprender com as imagens negativas. Imagens negativas é toda região que não está anotada, como fundo de cena e outros objetos

semelhantes que estão na imagem. Ele destaca que as razões dos falsos positivos se devem ao campo receptivo fixo nos detectores, que não muda conforme o tamanho do objeto. De acordo com Solawetz (2020), o comportamento de muitos falsos positivos na imagem pode ser causado por uma rede mal treinada.

Falsos negativos: não gerar uma caixa prevista para uma região anotada é considerado um falso negativo. Segundo (Goswami, 2020), para evitar falsos negativos deve-se treinar periodicamente o detector com novos exemplos de dados, pois as características do objeto a ser detectado devem estar presentes no *dataset* de treinamento. De acordo com os experimentos, apenas o SSD Mobilenet v2 FPNLite e o YOLOv4 não geraram falsos negativos.

Em relação as caixas delimitadoras mal posicionadas, segundo Girshick *et al* (2014); Lee, Kwak and Cho (2019), o motivo é que a rede neural aprende os deslocamentos das caixas de âncoras durante o treinamento.

Discussão sobre o tempo de treinamento

O tempo total de treinamento está relacionado com o número de iterações e o tempo de cada iteração. O tempo de cada iteração depende das condições e disponibilidade da GPU/CPU para processar uma imagem, da complexidade de cada arquitetura CNN e das características das imagens (quantidade de pixels). O ambiente em nuvem pode contribuir para variações no tempo de cada iteração.

A Tabela 19 mostra a síntese geral, do tempo médio de cada iteração e o tempo total de treinamento.

Método + CNN	**Iteração/Treinamento**	
Faster R-CNN Resnet 50	0,10955s	2h55m17s
Faster R-CNN Resnet 101	0,150s	4h00m49s
Faster Inception-Resnet-v2	0,417s	11h07m29s
SSD Mobilenet v2	0,093s	1h33m27s
SSD Mobilenet v2 FPNLite	**1,027s**	**3h25m25s**
SSD Resnet-50 v1 FPN	0,546s	9h06m12s
YOLOv2	4,215s	1h45m22s
YOLOv3	10,707s	4h27m40s
YOLOv4	**24,643s**	**10h16m04s**

Tabela 19 - Síntese do tempo de treinamento.

O SSD Mobilenet v2 FPNLite treinou em aproximadamente um terço do tempo do YOLOv4. Vale ressaltar que as inferências no SSD Mobilenet v2 FPNLite foram equivalentes ao YOLOv4.

Discussão sobre tempo de inferência

A Tabela 20 mostra na primeira coluna a arquitetura de detecção, seguido pelo tempo médio de cada inferência, o FPS e por último o tamanho do arquivo da rede neural. Essa última informação pode ser importante em algumas aplicações, como sistemas embarcados, por exemplo.

Método + CNN	**Inferência**	**FPS**	**Tamanho do arquivo da rede neural com os pesos treinados**
Faster R-CNN Resnet 50	0,110s	9	216MB
Faster R-CNN Resnet 101	0,125s	8	361MB
Faster Inception-Resnet-v2	0,378s	2	453MB
SSD Mobilenet v2	0,060s	16	35MB
SSD Mobilenet v2 FPNLite	0,071s	14	**19MB**
SSD Resnet-50 v1 FPN	0,084s	11	241MB
YOLOv2	**0,008s**	**125**	193MB
YOLOv3	0,017s	58	234MB
YOLOv4	0,032s	31	244MB

Tabela 20 - Síntese do tempo de inferência.

Em termos de velocidade, as arquiteturas YOLO são as mais rápidas, seguido pelo SSD e por último o Faster R-CNN.

No entanto, se considerarmos uma aplicação padrão com entrada de 30 FPS em tempo real, o SSD Mobilenet v2 FPNLite processaria uma imagem a cada dois frames aproximadamente e, ainda assim, seria uma aplicação em tempo real com eficiência. Segundo Ferreira (2022), aplicações com entrada acima de 30 FPS são utilizadas apenas em jogos e não é a principal escolha dos programadores em outras aplicações.

Discussão geral sobre os resultados

Os resultados apresentados neste capítulo superaram as expectativas com modelos de detecções que podem ser utilizados em aplicações com diversas condições, por exemplo:

- O caso em que se espera detecções menos robusta e que a velocidade seja prioridade, o YOLOv2 apresentou maior FPS;
- O caso que seja essencial o alto poder de detecção, então o SSD Mobilenet v2 FPNLite e o YOLOv4 demonstraram ser mais eficientes;

- O caso em que necessite treinar o modelo frequentemente e ainda necessite de detecções robustas, então o SSD Mobilenet v2 FPNLite é a melhor escolha;
- O caso em que o tamanho dos arquivos de pesos seja um desafio de implementação, por exemplo, em sistemas embarcados, então SSD Mobilenet v2 FPNLite pode ser mais eficiente.

Consolidação

Nas condições deste trabalho consolidamos dois modelos: o YOLOv4 e o SSD Mobilenet v2 FPNLite. As principais caracteristicas que os destacam são mostrados na Tabela 21.

	YOLOv4	SSD Mobilenet v2 FPNLite
mAP@50	99,95%	**100%**
FPS	**31**	14
Tempo de treinamento	Aprox. 10h	**Aprox. 3h**
Tamanho do arquivo	244 MB	**19 MB**

Tabela 21 - Consolidação.

Considerando a qualidade das detecções, o tempo de inferência e o tempo de treinamento, o SSD Mobilenet v2 FPNLite se sai melhor em pelo menos dois desses três quesitos. Como visto, enquanto o SSD Mobilenet v2 FPNLite detecta a cada dois frames, o YOLOv4 detecta em apenas um. Ainda assim, o SSD Mobilenet v2 FPNLite tem uma velocidade excelente para aplicações em tempo real.

Portanto, este trabalho adota o SSD Mobilenet v2 FPNLite como o detector de máscara mais eficiente dentre os nove modelos comparados. Adiante veremos uma comparação dos resultados deste trabalho com os resultados encontrados na literatura.

5.6 Resultados × Literatura

A Tabela 22 compara os resultados encontrados na literatura com os resultados obtidos neste projeto. Os dois trabalhos selecionados utilizam o mesmo *dataset* deste projeto; esse foi o principal cuidado na escolha para as comparações.

ID e Citação	**Parâmetros**				**Resultados**						
	Método + CNN	**Divisão treino e teste**	***Batch Size***	**Épocas**	**FPS**	**Tempo de treinamento**	**Medida de avaliação**				
							mAP	**Accuracy**	**F1-Score**	**Precision**	**Recall**
A3 – (Versiani et al., 2021)	SSD Mobilenet v2	Não informado	32	20	Não informado	Não informado		99%	99%		
A2 – (Nagrath et al., 2021)	SSD (Resnet-10) e Mobilenet v2	80/20	Não informado	100	15	Não informado		92,64%	93%		
Esta dissertação	**YOLOv4**	**80/20**	**64**	**75**	**31**	**10h16min**	**99,8%**				
Esta dissertação	**SSD Mobilenet v2 FPNLite**	**80/20**	**64**	**600**	**14**	**3h25min**	**100%**				

Tabela 22 - Resultados deste projeto × literatura.

Embora as métricas sejam diferentes, devido as propostas dos artigos, é possível observar a qualidade dos detectores apresentados na literatura. O estado da arte YOLOv4 utilizado neste trabalho e o SSD Mobilenet v2 FPNLite foram os detectores de destaque do projeto, que superam os resultados encontrados na literatura. A seguir será apresentado as considerações finais deste trabalho.

Capítulo 6 – Conclusão e Trabalhos Futuros

6.1 Conclusão

Neste trabalho foram apresentados os resultados dos algoritmos de detecção de objetos baseados em redes neurais Faster R-CNN, SSD e YOLO, aplicados em detecção de máscara. Foram comparados nove modelos pré-treinados em diferentes métricas, dentre eles o YOLOv4, que na revisão da literatura é conhecido como o estado da arte em detecção de objetos.

Embora este trabalho reconheça que o estado da arte é robusto e rápido em detecções, o tempo de treinamento e o tamanho do arquivo da rede neural podem ser um diferencial para determinadas aplicações. Nesta dissertação o detector mais eficiente é aquele que atendeu de forma satisfatória as métricas mAP@50, tempo de inferência e tempo de treinamento, que foi o SSD Mobilenet v2 FPNLite.

Comparando os nove modelos pré-treinados deste trabalho, o SSD Mobilenet v2 FPNLite apresentou detecções tão robustas quanto o YOLOv4. Foi constatado que o SSD Mobilenet v2 FPNLite tem uma velocidade de detecção suficiente para aplicações em tempo real, se comparado ao YOLOv4. O SSD Mobilenet v2 FPNLite apresentou o tempo de treinamento mais rápido que o YOLOv4. O arquivo final do SSD Mobilenet v2 FPNLite está em torno dos 19MB contra os 244MB do YOLOv4.

A revisão da literatura apresentou os principais detectores de objetos utilizados na detecção de máscara. A revisão mostrou as principais métricas, *datasets* utilizados nesse tipo de tarefa. Foi constatado os excelentes resultados dos algoritmos de detecção de objetos Faster R-CNN, SSD e YOLO na detecção de máscara e que foi comprovado nos experimentos desta dissertação. A revisão da literatura, contribuiu para identificar que o oclusão é um problema recorrente em detectores de objetos.

Finalmente, como resultado desta dissertação foi gerado e submetido à VISAPP 2023 (*InternationalConferenceon Computer Vision Theory and Applications*) o artigo com o título: "*Mask Recognition Using Object Detection Methods with Convolutional Neural Networks*".

Portanto, dentro das condições deste trabalho, os experimentos responderam à pergunta de pesquisa de qual é o método de detecção de máscara mais eficiente indicando a arquitetura SSD Mobilenet v2 FPNLite. O modelo atende os principais requisitos como taxa de acerto, velocidade de inferência e o tempo para treinar um

modelo. Este projeto de pesquisa não se limita apenas a detecção de máscara, a expectativa é que toda a metodologia para conclusão dos resultados possa ser aplicada a outros domínios na tarefa de detecção de objetos.

6.2 Trabalhos Futuros

Como forma a evoluir e dar continuidade ao desenvolvimento do presente trabalho em pesquisas futuras, algumas propostas são sugeridas:

- Assumindo que o problema de segmentação está resolvido. Como o problema de segmentação pode ser explorado na detecção de máscara ou qualquer outro objeto? As sugestões de ferramentas são: *Segment Anything* de abril 2023 pela empresa Meta AI (Kirillov et al., 2023). E o novo estado da arte YOLOv8, versão não oficial da empresa *Ultralytics* lançada em janeiro 2023 (Jocher, Chaurasia and Qiu, 2023).
- A detecção de máscara deste projeto apenas se propôs a identificar se a pessoa está ou não com máscara (rótulos “com_mascara” e “sem_mascara”). A sugestão é detectar o tipo de máscara que a pessoa está usando, com rótulos do tipo: “N95”, “PFF”, “Pano” etc.;
- A criação de um banco de imagens com os rótulos traduzidos maior do que o adotado neste trabalho, para que a comunidade se beneficie em futuros projetos;
- Oclusão é um problema em aberto na comunidade. Outra sugestão é explorar cenários complexos de oclusão com as arquiteturas deste projeto e o novo estado da arte.

Referências

Adhikarla, E. and Davison, B.D., 2021. Face Mask Detection on Real-World Webcam Images. In: *Proceedings of the Conference on Information Technology for Social Good*, GoodIT '21. [online] New York, NY, USA: Association for Computing Machinery. pp.139–144. https://doi.org/10.1145/3462203.3475903.

Ahmad, T., ma, Y., Yahya, M., Ahmad, B., Nazir, S., Haq, A. and Ali, R., 2020. Object Detection through Modified YOLO Neural Network. *Scientific Programming*. https://doi.org/10.1155/2020/8403262.

Alzubaidi, L., Santamaría, J., Manoufali, M., Mohammed, B., Fadhel, M.A., Zhang, J., Al-Timemy, A.H., Al-Shamma, O. and Duan, Y., 2021. *MedNet: Pre-trained Convolutional Neural Network Model for the Medical Imaging Tasks*. https://doi.org/10.48550/arXiv.2110.06512.

Androidkt, 2019. What is batch size, steps, iteration, and epoch in the neural network? *Knowledge Transfer*. Available at: <https://androidkt.com/batch-size-step-iteration-epoch-neural-network/> [Accessed 21 July 2022].

Approbato, B., 2021. *EPI: descubra qual máscara facial é a ideal para o seu trabalho!* [online] Apaixonados por Ferramentas. Available at: <https://apaixonadosporferramentas.com.br/epi-descubra-qual-mascara-facial-e-a-ideal-para-o-seu-trabalho/> [Accessed 5 October 2022].

ArcGIS, 2021. *How single-shot detector (SSD) works? | ArcGIS Developer*. [online] Available at: <https://developers.arcgis.com/python/guide/how-ssd-works/> [Accessed 25 November 2021].

Arcgis, 2022. *How Compute Accuracy For Object Detection works—ArcGIS Pro | Documentation*. [online] Available at: <https://pro.arcgis.com/en/pro-app/latest/tool-reference/image-analyst/how-compute-accuracy-for-object-detection-works.htm> [Accessed 5 November 2022].

Arriba, A. de, Oriol, M. and Franch, X., 2021. Applying Transfer Learning to Sentiment Analysis in Social Media. In: *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*. 2021 IEEE 29th International Requirements Engineering Conference Workshops (REW). pp.342–348. https://doi.org/10.1109/REW53955.2021.00060.

Bamne, B., Shrivastava, N., Parashar, L. and Singh, U., 2020. Transfer learning-based Object Detection by using Convolutional Neural Networks. In: *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*. 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC). pp.328–332. https://doi.org/10.1109/ICESC48915.2020.9156060.

Bengio, Y., 2016. *DeepLearning*. Cambridge, Massachusetts: MIT Press.

Bhandary, P., 2020. *prajnasb/observations*. Available at: <https://github.com/prajnasb/observations> [Accessed 13 August 2021].

Bhattarai, B., Raj Pandeya, Y. and Lee, J., 2021. Deep Learning-based Face Mask Detection Using Automated GUI for COVID-19. In: *2021 6th International Conference on Machine Learning Technologies*, ICMLT 2021. [online] New York, NY, USA: Association for Computing Machinery. pp.47–57. https://doi.org/10.1145/3468891.3468899.

Biswal, A., 2022. *Top 10 Deep Learning Algorithms You Should Know in 2023*. [online] Simplilearn.com. Available at: <https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm> [Accessed 22 October 2022].

Bochkovskiy, A., Wang, C.-Y. and Liao, H.-Y.M., 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv:2004.10934 [cs, eess]*. [online] Available at: <http://arxiv.org/abs/2004.10934> [Accessed 13 May 2021].

Bodla, N., Singh, B., Chellappa, R. and Davis, L.S., 2017. Soft-NMS -- Improving Object Detection With One Line of Code. *arXiv:1704.04503 [cs]*. [online] Available at: <http://arxiv.org/abs/1704.04503> [Accessed 24 April 2021].

Butantan, 2021. *Por que precisamos usar máscara para nos proteger contra a Covid-19?* [online] Available at: <https://butantan.gov.br/noticias/por-que-precisamos-usar-mascara-para-nos-proteger-contra-a-covid-19> [Accessed 22 December 2021].

Cabani, A., Hammoudi, K., Benhabiles, H. and Melkemi, M., 2021. MaskedFace-Net -- A Dataset of Correctly/Incorrectly Masked Face Images in the Context of COVID-19. *Smart Health*, 19, p.100144. https://doi.org/10.1016/j.smhl.2020.100144.

CHENG, B., 2019. Reducing false positives for object detection. *Reducing false positives for object detection*, p.52.

Chiba, S. and Sasaoka, H., 2021. Basic Study for Transfer Learning for Autonomous

Driving in Car Race of Model Car. In: *2021 6th International Conference on Business and Industrial Research (ICBIR)*. 2021 6th International Conference on Business and Industrial Research (ICBIR). pp.138–141. https://doi.org/10.1109/ICBIR52339.2021.9465856.

Christiansen, A., 2018. *Anchor Boxes — The key to quality object detection*. [online] Medium. Available at: <https://towardsdatascience.com/anchor-boxes-the-key-to-quality-object-detection-ddf9d612d4f9> [Accessed 4 November 2022].

COCO, 2022. *COCO - Common Objects in Context*. [online] Available at: <https://cocodataset.org/#detection-eval> [Accessed 20 April 2021].

Dey, S.K., Howlader, A. and Deb, C., 2021. MobileNet Mask: A Multi-phase Face Mask Detection Model to Prevent Person-To-Person Transmission of SARS-CoV-2. In: M.S. Kaiser, A. Bandyopadhyay, M. Mahmud and K. Ray, eds. *Proceedings of International Conference on Trends in Computational and Cognitive Engineering*, Advances in Intelligent Systems and Computing. Singapore: Springer. pp.603–613. https://doi.org/10.1007/978-981-33-4673-4_49.

Ding, Y., Li, Z. and Yastremsky, D., 2021. *Real-time Face Mask Detection in Video Data*. https://doi.org/10.48550/arXiv.2105.01816.

Ejaz, Md.S. and Islam, Md.R., 2019. Masked Face Recognition Using Convolutional Neural Network. In: *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*. 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI). pp.1–6. https://doi.org/10.1109/STI47673.2019.9068044.

Everingham, M., Eslami, S.M.A., Van Gool, L., Williams, C.K.I., Winn, J. and Zisserman, A., 2015. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1), pp.98–136. https://doi.org/10.1007/s11263-014-0733-5.

Ferreira, M.A., 2022. *30 ou 60 fps, a eterna discussão*. [online] PCManias.com. Available at: <https://www.pcmanias.com/30-ou-60-fps-a-eterna-discussao/> [Accessed 25 December 2022].

Filho, M., 2018. *As Métricas Mais Populares para Avaliar Modelos de Machine Learning*. [online] Available at: <https://mariofilho.com/as-metricas-mais-populares-para-avaliar-modelos-de-machine-learning/> [Accessed 7 November 2022].

Flickr-Faces, 2022. *NVlabs/ffhq-dataset*. Available at: <https://github.com/NVlabs/ffhq-dataset> [Accessed 25 November 2022].

Ge, S., Li, J., Ye, Q. and Luo, Z., 2022. *Masked Face Analysis*. [online] Available at: <https://imsg.ac.cn/research/maskedface.html> [Accessed 25 November 2022].

Ghiasi, G., Lin, T.-Y., Pang, R. and Le, Q.V., 2019. NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection. *arXiv:1904.07392 [cs]*. [online] Available at: <http://arxiv.org/abs/1904.07392> [Accessed 12 September 2021].

Girshick, R., 2015. Fast R-CNN. *arXiv:1504.08083 [cs]*. [online] Available at: <http://arxiv.org/abs/1504.08083> [Accessed 11 March 2021].

Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv:1311.2524 [cs]*. [online] Available at: <http://arxiv.org/abs/1311.2524> [Accessed 7 February 2021].

Gluon, 2022. *08. Finetune a pretrained detection model — gluoncv 0.11.0 documentation*. [online] Available at: <https://cv.gluon.ai/build/examples_detection/finetune_detection.html> [Accessed 29 October 2022].

Goswami, S., 2020. False Detection (Positives and Negatives) in Object Detection. *arXiv:2008.06986 [cs]*. [online] Available at: <http://arxiv.org/abs/2008.06986> [Accessed 17 October 2021].

Goyal, S., 2019. *MachineX: Image Data Augmentation Using Keras*. [online] Medium. Available at: <https://towardsdatascience.com/machinex-image-data-augmentation-using-keras-b459ef87cd22> [Accessed 22 October 2022].

He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*. [online] Available at: <http://arxiv.org/abs/1512.03385> [Accessed 3 May 2021].

Hindustantimes, 2020. *Automatic door opens only for those wearing mask. Watch viral video*. [online] Hindustan Times. Available at: <https://www.hindustantimes.com/it-s-viral/automatic-door-opens-only-for-those-wearing-mask-watch-viral-video/story-ALZ0c4Z2dpMAXr4jf0jgXO.html> [Accessed 23 December 2021].

Hofer, P., Roland, M., Schwarz, P., Schwaighofer, M. and Mayrhofer, R., 2021. Importance of different facial parts for face detection networks. In: *2021 IEEE*

International Workshop on Biometrics and Forensics (IWBF). 2021 IEEE International Workshop on Biometrics and Forensics (IWBF). pp.1–6. https://doi.org/10.1109/IWBF50991.2021.9465087.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861 [cs]*. [online] Available at: <http://arxiv.org/abs/1704.04861> [Accessed 22 August 2021].

Hui, J., 2017. *"Fast R-CNN and Faster R-CNN"*. [Blog] Jonathan Hui blog. Available at: <https://jhui.github.io/2017/03/15/Fast-R-CNN-and-Faster-R-CNN/> [Accessed 12 March 2021].

Ibitoye, O., 2021. A Brief Review of Convolutional Neural Network Techniques for Masked Face Recognition. In: *2021 IEEE Concurrent Processes Architectures and Embedded Systems Virtual Conference (COPA)*. 2021 IEEE Concurrent Processes Architectures and Embedded Systems Virtual Conference (COPA). pp.1–4. https://doi.org/10.1109/COPA51043.2021.9541448.

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z. and Qu, R., 2019. A Survey of Deep Learning-Based Object Detection. *IEEE Access*, 7, pp.128837–128868. https://doi.org/10.1109/ACCESS.2019.2939201.

Jocher, G., Chaurasia, A. and Qiu, J., 2023. *Introducing YOLOv8: Faster, Simpler, More Accurate*. Available at: <https://github.com/ultralytics/ultralytics> [Accessed 9 July 2023].

Kaggle, 2021. *Face Mask Detection*. [online] Available at: <https://kaggle.com/andrewmvd/face-mask-detection> [Accessed 15 August 2021].

Kamal, A., 2021. *YOLO, YOLOv2 and YOLOv3: All You want to know*. [online] Medium. Available at: <https://amrokamal-47691.medium.com/yolo-yolov2-and-yolov3-all-you-want-to-know-7e3e92dc4899> [Accessed 6 June 2021].

Kaur, J. and Singh, W., 2022. *Tools, techniques, datasets and application areas for object detection in an image: a review | SpringerLink*. [online] Available at: <https://link.springer.com/article/10.1007/s11042-022-13153-y> [Accessed 18 October 2022].

Khamlae, P., Sookhanaphibarn, K. and Choensawat, W., 2021. An Application of Deep-

Learning Techniques to Face Mask Detection During the COVID-19 Pandemic. In: *2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech)*. 2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech). pp.298–299. https://doi.org/10.1109/LifeTech52111.2021.9391922.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.-Y., Dollár, P. and Girshick, R., 2023. *Segment Anything*. https://doi.org/10.48550/arXiv.2304.02643.

Kumar, A., Kalia, A., Sharma, A. and Kaushal, M., 2021. A hybrid tiny YOLO v4-SPP module based improved face mask detection vision system. *Journal of Ambient Intelligence and Humanized Computing*. [online] https://doi.org/10.1007/s12652-021-03541-x.

Lee, S., Kwak, S. and Cho, M., 2019. Universal Bounding Box Regression and Its Applications. [online] Available at: <https://arxiv.org/abs/1904.06805v1> [Accessed 2 November 2021].

Li, C., Cao, J. and Zhang, X., 2020. Robust Deep Learning Method to Detect Face Masks. In: *Proceedings of the 2nd International Conference on Artificial Intelligence and Advanced Manufacture*, AIAM2020. [online] New York, NY, USA: Association for Computing Machinery. pp.74–77. https://doi.org/10.1145/3421766.3421768.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S., 2017. Feature Pyramid Networks for Object Detection. *arXiv:1612.03144 [cs]*. [online] Available at: <http://arxiv.org/abs/1612.03144> [Accessed 17 April 2021].

Liu, R. and Ren, Z., 2021. *Application of Yolo on Mask Detection Task*. https://doi.org/10.48550/arXiv.2102.05402.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A.C., 2016. SSD: Single Shot MultiBox Detector. *arXiv:1512.02325 [cs]*, 9905, pp.21–37. https://doi.org/10.1007/978-3-319-46448-0_2.

Loey, M., Manogaran, G., Taha, M.H.N. and Khalifa, N.E.M., 2021. Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection. *Sustainable Cities and Society*, 65, p.102600. https://doi.org/10.1016/j.scs.2020.102600.

Mahurkar, R.R. and Gadge, N.G., 2021. Real-time Covid-19 Face Mask Detection with

YOLOv4. In: *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*. 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC). pp.1250–1255. https://doi.org/10.1109/ICESC51422.2021.9533008.

Mandal, B., Okeukwu, A. and Theis, Y., 2021. *Masked Face Recognition using ResNet-50*. https://doi.org/10.48550/arXiv.2104.08997.

Martins, M. de F.M., 2018. *Estudos de Revisão de Literatura*. [online] Available at: <https://www.arca.fiocruz.br/bitstream/handle/icict/29213/Estudos_revisao.pdf;jsessionid=3276D110F598F62F145EB4FE04A2B39A?sequence=2> [Accessed 11 October 2022].

Mathworks, 2022. *Anchor Boxes for Object Detection - MATLAB & Simulink*. [online] Available at: <https://www.mathworks.com/help/vision/ug/anchor-boxes-for-object-detection.html> [Accessed 15 August 2022].

Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P. and Hemanth, J., 2021. SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2. *Sustainable Cities and Society*, 66, p.102692. https://doi.org/10.1016/j.scs.2020.102692.

Nithyashree, V., Roopashree, S., Duvvuri, A., Vanishree, L., Madival, D.A. and Vidyashree, G., 2021. A Solution to Covid-19: Detection and Recognition of Faces with Mask. In: *2021 International Conference on Intelligent Technologies (CONIT)*. 2021 International Conference on Intelligent Technologies (CONIT). pp.1–6. https://doi.org/10.1109/CONIT51480.2021.9498426.

Padilla, R., Lobato Passos, W., Dias, T., Netto, S. and da Silva, E., 2021. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics*, 10, pp.279–306. https://doi.org/10.3390/electronics10030279.

Padilla, R., Netto, S.L. and Silva, E.A.B. da, 2020. A Survey on Performance Metrics for Object-Detection Algorithms. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2020 International Conference on Systems, Signals and Image Processing (IWSSIP). pp.237–242. https://doi.org/10.1109/IWSSIP48289.2020.9145130.

Pan, S.J. and Yang, Q., 2010. A Survey on Transfer Learning. *IEEE Transactions on*

Knowledge and Data Engineering, 22(10), pp.1345–1359. https://doi.org/10.1109/TKDE.2009.191.

Razavi, M., Alikhani, H., Janfaza, V., Sadeghi, B. and Alikhani, E., 2021. An Automatic System to Monitor the Physical Distance and Face Mask Wearing of Construction Workers in COVID-19 Pandemic. *arXiv:2101.01373 [cs]*. [online] Available at: <http://arxiv.org/abs/2101.01373> [Accessed 9 May 2022].

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You Only Look Once: Unified, Real-Time Object Detection. *arXiv:1506.02640 [cs]*. [online] Available at: <http://arxiv.org/abs/1506.02640> [Accessed 13 May 2021].

Redmon, J. and Farhadi, A., 2017. YOLO9000: Better, Faster, Stronger. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp.6517–6525. https://doi.org/10.1109/CVPR.2017.690.

Redmon, J. and Farhadi, A., 2018. YOLOv3: An Incremental Improvement. *arXiv:1804.02767 [cs]*. [online] Available at: <http://arxiv.org/abs/1804.02767> [Accessed 13 May 2021].

Ren, S., He, K., Girshick, R. and Sun, J., 2016. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv:1506.01497 [cs]*. [online] Available at: <http://arxiv.org/abs/1506.01497> [Accessed 7 February 2021].

Rusli, M.H., Sjarif, N.N.A., Yuhaniz, S.S., Kok, S. and Kadir, M.S., 2021. Evaluating the Masked and Unmasked Face with LeNet Algorithm. In: *2021 IEEE 17th International Colloquium on Signal Processing Its Applications (CSPA)*. 2021 IEEE 17th International Colloquium on Signal Processing Its Applications (CSPA). pp.171–176. https://doi.org/10.1109/CSPA52141.2021.9377283.

Sakthimani, S., Harikrishna, R., Prakash, R.K. and Piranesh, S., 2021. FACE MASK DETECTION AND AUTOMATIC DOOR OPENER USING YOLO. *Türkiye Fizyoterapistler Derneği*, p.5.

Sanjaya, S.A. and Adi Rakhmawan, S., 2020. Face Mask Detection Using MobileNetV2 in The Era of COVID-19 Pandemic. In: *2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI)*. 2020 International Conference on Data Analytics for Business and Industry: Way Towards a

Sustainable Economy (ICDABI). pp.1–5. https://doi.org/10.1109/ICDABI51230.2020.9325631.

Santos, L., 2020. *Object Localization and Detection*. [online] Available at: <https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/deep_learning/object_localization_and_detection> [Accessed 23 March 2021].

Shaha, M. and Pawar, M., 2018. Transfer Learning for Image Classification. In: *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA). pp.656–660. https://doi.org/10.1109/ICECA.2018.8474802.

Shao, L., Zhu, F. and Li, X., 2015. Transfer Learning for Visual Categorization: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 26(5), pp.1019–1034. https://doi.org/10.1109/TNNLS.2014.2330900.

Shen, K., 2018. Effect of batch size on training dynamics. *Mini Distill*. Available at: <https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e> [Accessed 31 October 2021].

Shorten, C. and Khoshgoftaar, T.M., 2019. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1), p.60. https://doi.org/10.1186/s40537-019-0197-0.

Shuoyang1213, 2022. *WIDER FACE: A Face Detection Benchmark*. [online] Available at: <http://shuoyang1213.me/WIDERFACE/> [Accessed 25 November 2022].

Singh, S., Ahuja, U., Kumar, M., Kumar, K. and Sachdeva, M., 2021. Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment. *Multimedia Tools and Applications*, 80(13), pp.19753–19768. https://doi.org/10.1007/s11042-021-10711-8.

Solawetz, J., 2020. *What are Anchor Boxes in Object Detection?* [online] Roboflow Blog. Available at: <https://blog.roboflow.com/what-is-an-anchor-box/> [Accessed 22 October 2021].

Srinivasan, S., Rujula Singh, R., Biradar, R.R. and Revathi, S., 2021. COVID-19 Monitoring System using Social Distancing and Face Mask Detection on Surveillance

video datasets. In: *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*. 2021 International Conference on Emerging Smart Computing and Informatics (ESCI). pp.449–455. https://doi.org/10.1109/ESCI50559.2021.9396783.

Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A., 2016. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. [online] Available at: <https://arxiv.org/abs/1602.07261v2> [Accessed 29 August 2021].

TensorFlow, 2021. *TensorFlow*. [online] Available at: <https://www.tensorflow.org/?hl=pt-br> [Accessed 18 August 2021].

TensorFlow 2 Object Detection API, 2021. *Installation — TensorFlow 2 Object Detection API tutorial documentation*. [online] Available at: <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html#set-env> [Accessed 21 August 2021].

Uijlings, J.R.R., Sande, K.E.A. van de, Gevers, T. and Smeulders, A.W.M., 2013. Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2), pp.154–171. https://doi.org/10.1007/s11263-013-0620-5.

Varshini, B., Yogesh, H., Pasha, S.D., Suhail, M., Madhumitha, V. and Sasi, A., 2021. IoT-Enabled smart doors for monitoring body temperature and face mask detection. *Global Transitions Proceedings*, 2(2), pp.246–254. https://doi.org/10.1016/j.gltp.2021.08.071.

Venkateswarlu, I.B., Kakarla, J. and Prakash, S., 2020. Face mask detection using MobileNet and Global Pooling Block. In: *2020 IEEE 4th Conference on Information Communication Technology (CICT)*. 2020 IEEE 4th Conference on Information Communication Technology (CICT). pp.1–5. https://doi.org/10.1109/CICT51604.2020.9312083.

Vibhuti, Jindal, N., Singh, H. and Rana, P.S., 2022. Face mask detection in COVID-19: a strategic review. *Multimedia Tools and Applications*, 81(28), pp.40013–40042. https://doi.org/10.1007/s11042-022-12999-6.

Vinh, T.Q. and Anh, N.T.N., 2020. Real-Time Face Mask Detector Using YOLOv3 Algorithm and Haar Cascade Classifier. In: *2020 International Conference on Advanced Computing and Applications (ACOMP)*. 2020 International Conference on Advanced Computing and Applications (ACOMP). pp.146–149. https://doi.org/10.1109/ACOMP50827.2020.00029.

Wang, B., Zhao, Y. and Chen, C.L.P., 2021. Hybrid Transfer Learning and Broad Learning System for Wearing Mask Detection in the COVID-19 Era. *IEEE Transactions on Instrumentation and Measurement*, 70, pp.1–12. https://doi.org/10.1109/TIM.2021.3069844.

Wang, B., Zheng, J. and Chen, C.L.P., 2022. A Survey on Masked Facial Detection Methods and Datasets for Fighting Against COVID-19. *IEEE Transactions on Artificial Intelligence*, 3(3), pp.323–343. https://doi.org/10.1109/TAI.2021.3139058.

Wang, C.-Y., Liao, H.-Y.M., Yeh, I.-H., Wu, Y.-H., Chen, P.-Y. and Hsieh, J.-W., 2019. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. *arXiv:1911.11929 [cs]*. [online] Available at: <http://arxiv.org/abs/1911.11929> [Accessed 18 June 2021].

Wang, Z., Wang, P., Louis, P.C., Wheless, L.E. and Huo, Y., 2021. *WearMask: Fast In-browser Face Mask Detection with Serverless Edge Computing for COVID-19*. https://doi.org/10.48550/arXiv.2101.00784.

Wu, P., Li, H., Zeng, N. and Li, F., 2022. FMD-Yolo: An efficient face mask detection method for COVID-19 prevention and control in public. *Image and Vision Computing*, 117, p.104341. https://doi.org/10.1016/j.imavis.2021.104341.

Xu, M., Wang, H., Yang, S. and Li, R., 2020. Mask wearing detection method based on SSD-Mask algorithm. In: *2020 International Conference on Computer Science and Management Technology (ICCSMT)*. 2020 International Conference on Computer Science and Management Technology (ICCSMT). pp.138–143. https://doi.org/10.1109/ICCSMT51754.2020.00034.

X-zhangyang, 2022. *Real-World Masked Face Dataset, RMFD*. Available at: <https://github.com/X-zhangyang/Real-World-Masked-Face-Dataset> [Accessed 25 November 2022].

Yosinski, J., Clune, J., Bengio, Y. and Lipson, H., 2014. How transferable are features in deep neural networks? *arXiv:1411.1792 [cs]*. [online] Available at: <http://arxiv.org/abs/1411.1792> [Accessed 14 April 2021].

Zhao, Z.-Q., Zheng, P., Xu, S. and Wu, X., 2019. Object Detection with Deep Learning: A Review. *arXiv:1807.05511 [cs]*. [online] Available at: <http://arxiv.org/abs/1807.05511> [Accessed 19 February 2021].

Apêndices

Apêndice 1 – Base de Imagens com Máscara

Apresentado na figura A-1,imagens de pessoas com máscaras do banco de dados original. Nota-se que as foram criadas sinteticamente.

Figura A-1 -Rostos com máscara.

Apresentado na figura A-3, a técnica de aumento de dados presente na categoria “com_mascara”, do banco de dados original.

Figura A-2 -Aumento de dados na categoria com máscara.

Apêndice 2 – Base de Imagens sem Máscara

Apresentado na figura A-3, imagens de pessoas sem máscaras do banco de dados original.

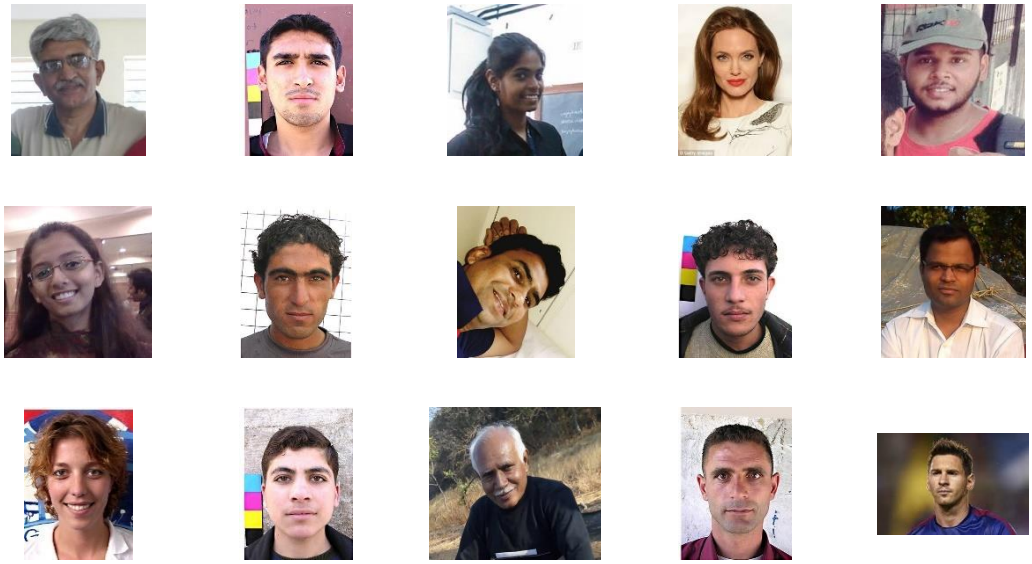


Figura A-3 -Rostos sem máscaras.

Apresentado na figura A-4, a técnica de aumento de dados presente na categoria “sem_mascara”, do banco de dados original.

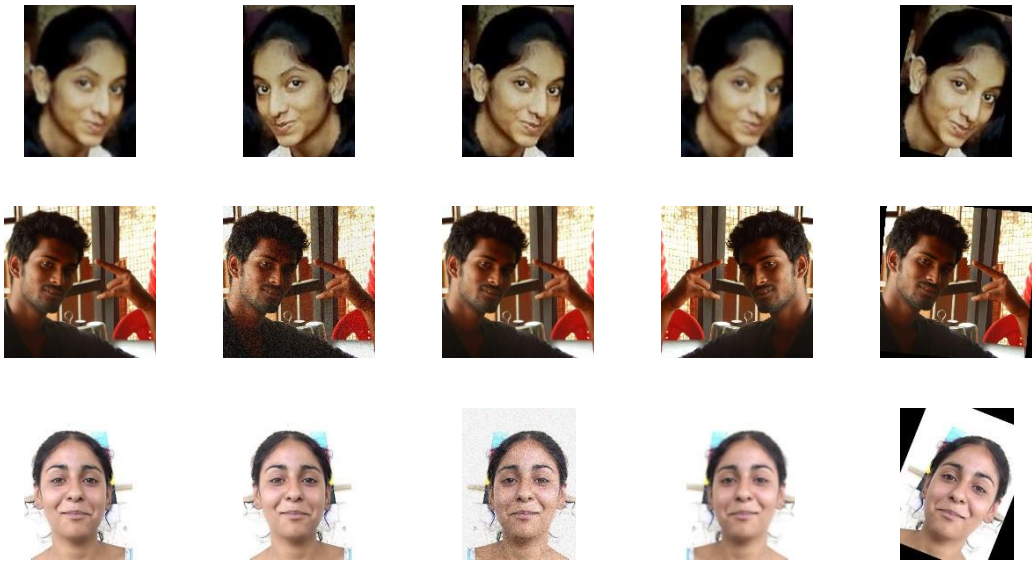


Figura A-4 -Aumento de dados na categoria sem máscara.

Apêndice 3 – Script para Inferência - *Tensorflow API Object Detection*

Depois de carregado o modelo na memória foi adicionado ao *script* a função "*time.time()*". A alteração está destacada em amarelo e calcula o tempo de inferência. No Exemplo do terminal do Google Colaboratory:

```
#CARREGAMENTO DO CAMINHO DA IMAGEM
import os
IMAGE_PATHS = ""
# Usando a imagem m11.JPG como teste de exemplo
IMAGE_PATHS = '/content/drive/MyDrive/tensorflow/image m/m11.JPG'
IMAGE_PATHS

#CARREGAMENTO COMPLETO DO MODELO

%cd /content
import time
import tensorflow as tf # Added as colab instance often crash
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_util s
# Caminho do label
PATH_TO_LABELS = '/content/drive/MyDrive/tensorflow/wo rkspace/training_demo/annotations/label_map.pbtxt'
# Caminho do modelo salvo
PATH_TO_SAVED_MODEL = '/content/drive/MyDrive/tenso rflow/workspace/training_demo/exported-models/1_faster_rcnn_Inception_resnet_v2_640/saved_model'
print('Loading model...', end='')
start_time = time.time()
# Carrega o modelo e constrói a função de detecção
detect_fn = tf.saved_model.load(PATH_TO_SAVED_MODEL )
end_time = time.time()
elapsed_time = end_time - start_time
print('Done! Took {} seconds'.format(elapsed_time))
# Conjunto de índices das categorias
category_index = label_map_util.create_category_index_from_ labelmap(PATH_TO_LABELS,use_display_name=True)

#INFERÊNCIA DE UMA IMAGEM

import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

start_time = time.time()  #Inicia o tempo de processamento

#Inseri uma imagem para um numpyarray
image_np = np.array(Image.open(IMAGE_PATHS))
# converte em um tensor
input_tensor = tf.convert_to_tensor(image_np)
#O modelo espera um batch de imagens
input_tensor = input_tensor[tf.newaxis, ...]
# Input_tensor = np.expand_dims(image_np, 0)
detections = detect_fn(input_tensor)
# Todas as saídas são tensores de lotes.
# Converta em arraysnumpy e pegue o índice [0] para remover a dimensão do lote.
# Estamos interessados apenas nos primeiros num_detections.
num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
               for key, value in detections.items()}
detections['num_detections'] = num_detections

# Detecção_classes deve ser ints.
detections['detection_classes'] = detections['detection_classes'].
astype(np.int64)
image_np_with_detections = image_np.copy()
viz_utils.visualize_boxes_and_labels_on_image_array(
      image_np_with_detections,
      detections['detection_boxes'],
      detections['detection_classes'],
      detections['detection_scores'],
      category_index,
      use_normalized_coordinates=True,
      max_boxes_to_draw=20,
      min_score_thresh=.50,
      agnostic_mode=False)

plt.figure(figsize = (12,8))
plt.imshow(image_np_with_detections)

end_time = time.time() #Fim do tempo de processamento

elapsed_time = end_time - start_timeCalcula o tempo
print('Done! Took {} seconds'.format(elapsed_time))
print('Done')
plt.show()
```