UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

LUCAS FREIRE FERREIRA

DETECÇÃO DE FACES: UMA COMPARAÇÃO ENTRE VIOLA-JONES E YOLO

VITÓRIA-ES

MARÇO/2022

Lucas Freire Ferreira

DETECÇÃO DE FACES: UMA COMPARAÇÃO ENTRE VIOLA-JONES E YOLO

Parte manuscrita do Projeto de Graduação do aluno Lucas Freire Ferreira, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Vitória-ES

Março/2022

Lucas Freire Ferreira

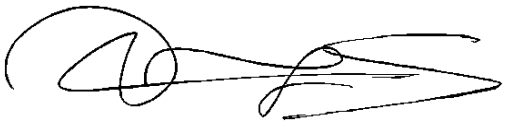
DETECÇÃO DE FACES: UMA COMPARAÇÃO ENTRE VIOLA-JONES E YOLO

Parte manuscrita do Projeto de Graduação do aluno Lucas Freire Ferreira, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovado em 25 de março de 2022.

COMISSÃO EXAMINADORA:

Prof. Dr. Patrick Marques Ciarelli
Universidade Federal do Espírito Santo
Orientador



Prof. Dr. Jorge Leonid Aching Samatelo
Universidade Federal do Espírito Santo
Examinador

Erick Ramos dos Santos

Eng. Erick Ramos dos Santos
Examinador

Vitória-ES

Março/2022

As pessoas que continuaram acreditando.

AGRADECIMENTOS

Gostaria de agradecer primeiramente à Deus por ter me dado forças, determinação e saúde para superar as adversidades, concluir o curso e este trabalho. Agradeço à minha família por todo o amor e incentivo durante minha vida. Agradeço em especial aos meus pais e minha irmã por nunca ter faltado uma palavra de apoio. Agradeço à essa instituição por ter me proporcionado as experiências e conhecimentos necessários para me tornar bacharel em engenharia elétrica. Agradeço aos docentes que me acompanharam durante todo o curso, sempre dispostos a me ensinar e esclarecer as dúvidas. Agradeço em especial ao meu orientador, Patrick Marques Ciarelli, pela orientação, apoio, ajuda, correções e principalmente pela paciência, sem ele, nada disso seria possível. À banca examinadora pela aceitação do convite e pelo tempo investido para leitura e avaliação desse trabalho. Agradeço a todos aqueles que tiveram influência direta ou indireta em minha formação.

RESUMO

Detecção facial apesar de não ser um tópico recente, é muito discutido atualmente devido ao seu alto grau de aplicação, seja em áreas como a de vídeo monitoramento ou até mesmo como a primeira etapa em sistemas de reconhecimento de faces. Ao longo do tempo foram desenvolvidas várias técnicas computacionais cada vez mais avançadas para executar essa tarefa, sendo as mais atuais baseadas em redes de aprendizado profundo. O avanço e criação de novas técnicas traz diversas novas possibilidades, porém ainda não existe uma solução única para todos os problemas e aplicações. O presente projeto de graduação tem como foco comparar uma técnica clássica e uma técnica mais atual, que utiliza aprendizado profundo. Para essa tarefa são utilizados dois conjuntos de dados públicos: o *Wider Faces* e o *Face Detection Data Set and Benchmark*. A clássica técnica Viola-Jones, ainda muito utilizada em celulares e dispositivos com menor poder computacional, foi comparada com a *You Only Look Once* (YOLO), representante de uma técnica recente de aprendizado profundo. As duas foram selecionadas por serem conhecidas como técnicas rápidas que podem ser utilizadas em sistemas em tempo real. Para a realização deste trabalho, um estudo da arquitetura e funcionamento de cada uma dessas técnicas foi feito, apresentando uma descrição detalhada de seus funcionamentos internos. Foi também necessário realizar o treinamento de uma rede YOLOv5 para classificação facial. Após realizar as predições, os resultados obtidos foram: (*i*) Em todos os casos de acurácia a técnica YOLO se mostrou melhor, obtendo $85,49\%$ para o conjunto de dados *Face Detection Data Set and Benchmark* e $56,55\%$ para a *Wider Faces*; (*ii*) No aspecto eficiência computacional foram obtidos resultados mistos, com 33 FPS para a Viola-Jones e 100 FPS para a YOLO em seus melhores casos.

Palavras-chave: Aprendizado profundo; Redes neurais convolucionais; YOLO; Viola-Jones; Detecção facial.

ABSTRACT

Face detection, despite not being a recent topic, is still widely discussed due to its high degree of application, whether in areas such as video monitoring or even as the first step in face recognition systems. Over time, several increasingly advanced computational techniques have been developed to perform this task, the most modern ones being based on deep learning networks. The advancement and creation of new techniques brings numerous new possibilities but still there is not only one solution to all problems and applications. This project focuses on making a comparison between a classic technique and a more modern technique, which uses deep learning. For this comparison, two public datasets were used: Wider Faces and Face Detection Data Set and Benchmark. The classic Viola-Jones technique, still widely used in cell phones and devices with less computational power, was compared with the You Only Look Once (YOLO), representative of a recent technique based on deep learning. Both were selected because they are known as fast techniques that can be used in real-time systems. For this assignment, a study of the architecture of each technique was done. It was also required to train a YOLOv5 network for facial classification. After the predictions, the results were: (*i*) In all accuracy cases the YOLO technique was better, obtaining $86,38\%$ for the Face Detection Data Set and Benchmark and $67,97\%$ for the Wider Face; (*ii*) In the computational efficiency aspect the results were mixed, with 33 FPS for Viola-Jones and 100 FPS for YOLO in the best case scenario for each.

Keywords: Deep learning; Convolutional neural networks; YOLO; Viola-Jones; Face detection.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CNN	*Convolutional Neural Network*
CSP	*Cross Stage Partial Networks*
DL	*Deep Learning*
FPS	*Frames por Segundo*
FDDB	*Face Detection Data Set and Benchmark*
IoU	*Intersection over Union*
ML	*Machine Learning*
px	*pixel*
PANet	*Path Aggregation Network*
RNC	*Rede Neural Convolucional*
SVM	*Support Vector Machine*
UFES	Universidade Federal do Espírito Santo
YOLO	*You Only Look Once*

SUMÁRIO

1 INTRODUÇÃO

1.1 Apresentação

O ser humano é um ser social e gerir uma sociedade é uma tarefa relativamente simples em um pequeno grupo de amigos ou até mesmo uma vila de poucos habitantes, entretanto, em uma sociedade grande e complexa, existe a necessidade de uma melhor gestão e administração das relações entre humanos e ambiente (ZIMMERMANN, 2017). Dessa necessidade, surgiram métodos automatizados para lidar com situações cotidianas, dentre esses um muito importante é a de detecção de faces.

A detecção automática de faces é uma importante tarefa na área de visão computacional, sendo o primeiro passo e o mais importante na utilização de imagens de faces. Aplicações como automação de segurança residencial (AYDIN; OTHMAN, 2017), interação homem-máquina (BARTLETT et al., 2003) e tecnologias assistivas (ISMAIL et al., 2011), são apenas alguns exemplos de uma área que vem sendo abordada por um número cada vez maior de pesquisas.

Apesar de todo o trabalho realizado em cima da detecção de faces, esse ainda não é um problema ainda totalmente resolvido e diversas técnicas vêm sendo desenvolvidas ao longo do tempo com suas respectivas vantagens e desvantagens. O foco deste trabalho é realizar uma comparação entre duas abordagens de detecção de faces: a técnica Viola-Jones (VIOLA; JONES, 2001), uma técnica clássica que ainda é muito utilizada em dispositivos móveis com pouco poder de processamento devido a sua velocidade e precisão, e a *You Only Look Once* (YOLO)(REDMON et al., 2016), um detector relativamente recente de objetos que utiliza redes neurais profundas. A proposta é analisar diferenças de desempenho entre elas e identificar em quais condições técnicas recentes como a YOLO podem ser melhor do que técnicas clássicas como Viola-Jones. Embora existam técnicas mais poderosas do que a YOLO, como a *TinaFace* (ZHU et al., 2020), *AInnoFace* (ZHANG et al., 2019) e a *Face R-FCN* (WANG et al., 2017), ela foi escolhida por ser rápida o suficiente para ser aplicada em detecção em tempo (quase) real, mesmo com um hardware modesto.

1.2 Justificativa

Com o surgimento de novas tecnologias, tem-se métodos cada vez mais efetivos e amigáveis de interação homem-máquina que não dependem mais de ferramentas tradicionais como mouse e teclado. Associado a isso, o barateamento do custo para aquisição de vídeo e

processamento tornam viáveis e mais atraentes as aplicações que dependem de detecção facial (RIZVI, 2011).

A detecção de faces é uma tecnologia que, dada uma imagem, permite determinar a existência e localização de rostos presentes nela. Apesar dessa ser uma tarefa simples para um humano resolver, para uma máquina essa não é uma tarefa trivial, fatores como pose, variações estruturais do rosto, expressão, oclusão e iluminação são apenas alguns dos desafios associados a essa tarefa.

Dessa forma, neste projeto busca-se aprofundar conhecimentos e estudar os métodos Viola-Jones e YOLO. Procura-se também realizar comparações entre esses métodos de reconhecimento utilizando dois conjuntos de dados de forma a entender melhor suas aplicabilidades e limitações.

1.3 Objetivos

Objetivo Geral

Realizar o estudo e comparação de duas técnicas que podem ser utilizadas para detecção facial em dois conjuntos de dados públicos de imagens. A primeira foi proposta em Viola e Jones (2001), conhecida como Viola-Jones, e a segunda é a técnica proposta por Joe Redmon (2015), conhecida como *You Only Look Once* (YOLO), em sua versão mais recente, a YOLOv5, proposta por Glenn Jocher (2020).

Objetivos Específicos

- Revisar e empregar a detecção de faces utilizando Viola-Jones;
- Revisar e empregar a detecção de faces utilizando YOLO;
- Comparar ambas as técnicas utilizando os conjuntos de dados *Face Detection Data Set and Benchmark* (JAIN; LEARNED-MILLER, 2010) e *Wider Face* (YANG et al., 2016).

1.4 Estrutura do Texto

O presente trabalho está estruturado em capítulos, de forma que este capítulo inicial tem como objetivo contextualizar a problemática e justificar a importância do estudo realizado neste trabalho. Ainda, neste capítulo, são apresentados os objetivos do trabalho e a estrutura textual da proposta de projeto.

Em seguida, no Capítulo 2, são explanados os conceitos fundamentais e conhecimentos necessários para a compreensão do estudo a ser realizado em conjunto com o referencial teórico. São apresentadas noções de Reconhecimento Facial, Redes Neurais Profundas e Convolucionais, Viola-Jones e YOLO.

No Capítulo 3 são apresentadas as etapas da metodologia da pesquisa, as métricas de avaliação e as informações dos conjuntos de dados.

No Capítulo 4 são expostos os resultados e análises da pesquisa e, por fim, no Capítulo 5 são apresentadas as conclusões e propostas de melhorias para pesquisas futuras.

2 EMBASAMENTO TEÓRICO

Neste capítulo será feita uma breve descrição sobre detecção facial e os diferentes grupos de métodos existentes. A seguir, serão explicados os dois métodos escolhidos para a utilização no projeto e seus princípios.

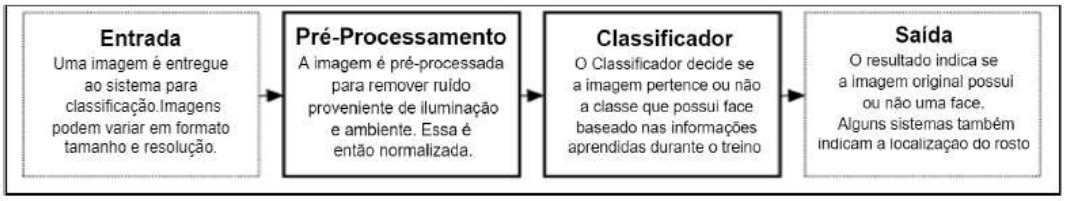
2.1 Detecção Facial

O objetivo da detecção facial é identificar, dada uma imagem, se existe ou não um rosto naquela imagem e, na maioria dos casos, sua localização. Apesar de ser uma tarefa simples para um cérebro humano, existem diversas dificuldades que tornam esse processo não trivial como *(i)* Expressões faciais, *(ii)* Resolução da imagem, *(iii)* Iluminação, *(iv)* Orientação da face e *(v)* oclusão.

Como pode ser observado na Figura 1, um sistema de detecção facial pode ser composto por 4 etapas distintas. A primeira delas representa a entrada do sistema, podendo ser tanto uma imagem quanto um vídeo. A segunda etapa é o pré-processamento da entrada. Existem diversas técnicas que são utilizadas para facilitar e aumentar a precisão da detecção, algumas delas são: (*i*) remoção de ruído, (*ii*) tratamento de iluminação, (*iii*) equalização de histograma. A terceira etapa é a etapa de classificação onde é utilizado alguma metodologia encontrar o objeto desejado na entrada. Por fim, a quarta e última etapa é a saída do sistema, onde pode-se ter apenas um resultado positivo ou negativo para a presença de um rosto ou então é apresentada a localização do rosto na imagem ou vídeo.

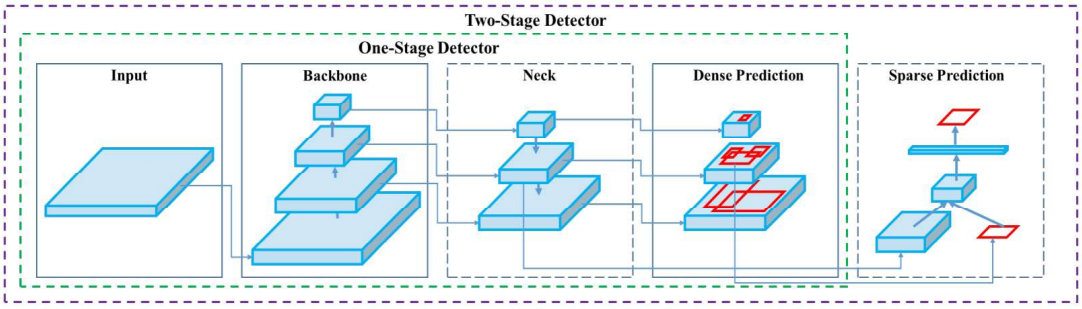
O foco deste trabalho será em cima de dois métodos, um baseado em características, representado pelo algoritmo Viola-Jones, e um baseado em aparência, representado pelo detector de objetos YOLO. Vale ressaltar também que o método Viola-Jones se encaixa como um classificador no sistema genérico, enquanto a YOLO é um detector de objetos completo e é melhor representado pela Figura 2. A Figura 2 apresenta a arquitetura genérica de detectores de objeto com uma ou duas etapas. Esses podem ser divididos em algumas etapas: *(i)* a entrada do sistema; *(ii) Backbone* onde o modelo extrai *features* da entrada; *(iii) Neck* onde as *features* extraídas são agregadas e misturadas e combinadas e *(iv) Head* onde é realizada a etapa de detecção (THUAN, 2021).

Figura 1 – Sistema de detecção facial genérico



Adaptado de AL-Allaf (2014).

Figura 2 – Sistema de detecção de objetos com uma ou duas etapas



Fonte: Bochkovskiy, Wang e Liao (2020).

Classificadores baseados em características buscam atributos invariáveis de faces para realizar a detecção. A premissa é baseada na observação de que humanos podem detectar rostos sem esforço e objetos em diferentes poses e condições de iluminação, então devem existir propriedades que são invariantes sobre essas variabilidades. Características como cor da pele, sobrancelhas, olhos, nariz, boca e linha do cabelo são comumente extraídas usando técnicas, por exemplo, detectores de borda. Com base nas características extraídas, um modelo estatístico é construído para verificar a existência de um rosto (KUMAR; KAUR; KUMAR, 2018).

Já os classificadores baseados em aparência dependem de técnicas de análise estatística e aprendizado de máquina para encontrar características relevantes das imagens que contêm faces. As características aprendidas são na forma de modelos de distribuição ou funções discriminantes que são consequentemente usados para detecção de rosto (KUMAR; KAUR; KUMAR, 2018).

2.2 Viola-Jones

Desenvolvido por Paul Viola e Michael Jones em 2001, o *framework* para detecção de objetos Viola-Jones consegue detectar de forma rápida e com alta precisão objetos em imagens e funciona muito bem para faces devido a utilização de atributos Haar. Apesar de

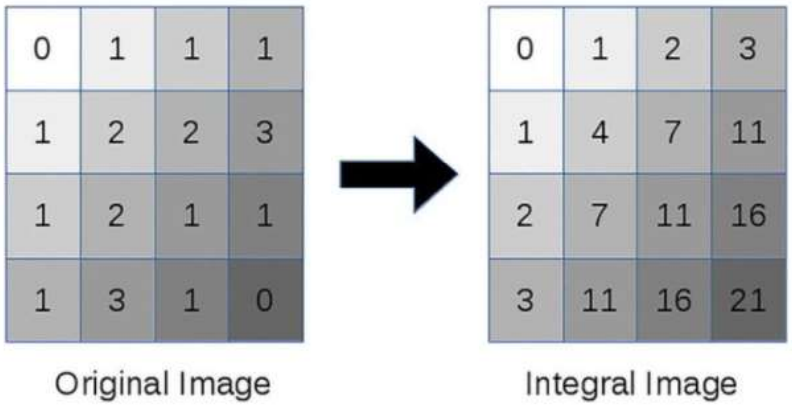
não ser um método recente, esse *framework* ainda é um dos mais utilizados, junto com algumas redes neurais convolucionais (CNN), para a detecção de face (LEE, 2020). Para atingir esse sucesso, o *framework* combina diferentes métodos e conceitos, sendo eles: (*i*) Imagem integral, (*ii*) Atributos Haar, (*iii*) Algoritimo AdaBoost e (*iv*) Classificadores em cascata, que serão explicados nas próximas subseções.

2.2.1 Imagem integral

A Imagem Integral, também conhecida como tabela de soma de áreas, é aplicada na visão computacional para tornar mais rápido o cálculo recorrente da soma dos valores dos pixels em uma área. Para isso, é pré-computada uma nova imagem usando a original, onde o valor de um pixel é igual a soma dos valores dos pixels acima e a esquerda dele (ele mesmo incluso) (SZELISKI, 2011), como demonstrado na Figura 3.

Esse conceito é muito importante para o *framework* por permitir o cálculo de atributos retangulares de forma muito rápida (VIOLA; JONES, 2001), pois só é necessário o valor de 4 pontos para o cálculo da intensidade da área, como mostra a Figura 4.

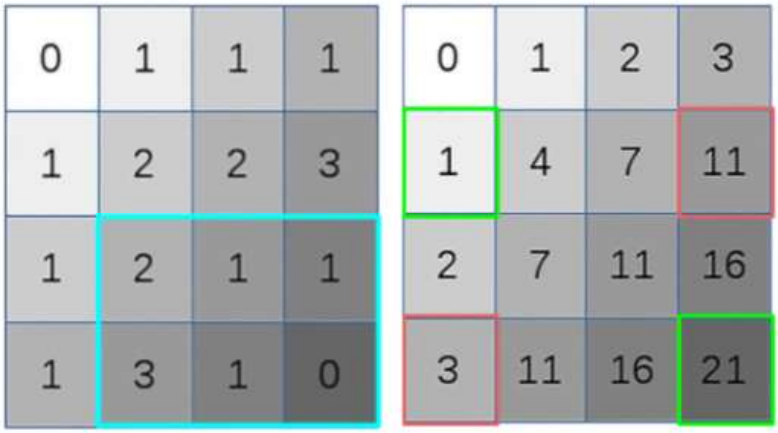
Figura 3 – Transformação para imagem integral



Fonte:Mujtaba (2020).

Para entender melhor a relevância desse cálculo rápido é importante entender como funciona a detecção de atributos em uma imagem utilizando a chamada janela deslizante. Isso se dá utilizando um filtro (*kernel*), no caso do método Viola-Jones são os atributos Haar, que desliza sobre a imagem buscando regiões de interesse, gerando várias sub-janelas candidatas. O classificador então avalia as sub-janelas.

Figura 4 – Cálculo da soma da intensidade da área marcada usando os valores destacados. Soma-se os verdes e são subtraidos os vermelhos.

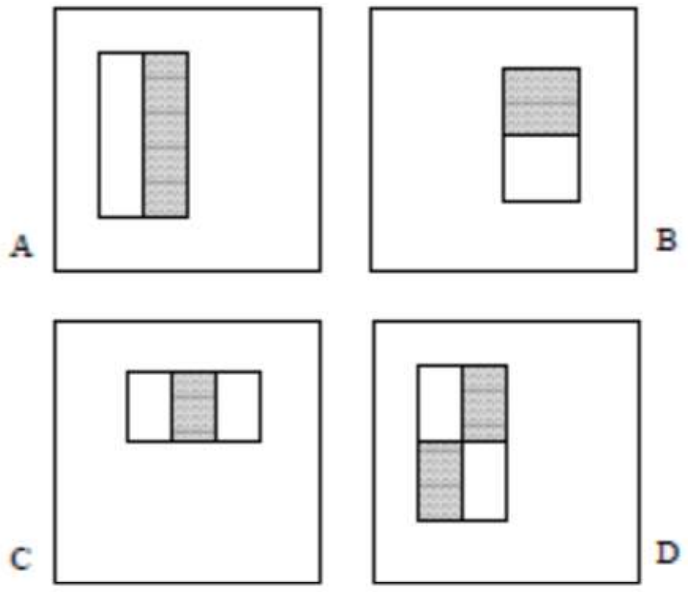


Fonte:Mujtaba (2020).

2.2.2 Atributos Haar

Um atributo Haar consiste em uma área escura e uma área clara que retornam o valor da soma dos pixels na área clara subtraído da soma dos pixels presentes na área escura. O *framework* utiliza 4 tipos diferentes de atributos, apresentados na Figura 5, para o procedimento de classificação, tirando vantagem da maior velocidade de processamento devido ao seu formato retangular.

Figura 5 – Exemplo dos 4 tipos de atributos utilizados em comparação com a janela de detecção.



Fonte: Viola e Jones (2001).

2.2.3 *Boosting* Adaptativo (*AdaBoost*)

O *framework* Viola-Jones introduziu na comunidade de visão computacional o conceito de *boosting*, que consiste em um método genérico de melhorar a precisão de um algoritimo de aprendizado. *AdaBoost* resume-se em gerar um classificador forte usando a soma de vários *weak learners* (classificadores fracos) que em *Machine Learning* são algoritmos que possuem precisão um pouco melhor do que adivinhar ao acaso e, na maioria das variantes de *boosting*, são funções de limiar (árvores de decisão de apenas duas folhas e um nó). (SZELISKI, 2011).

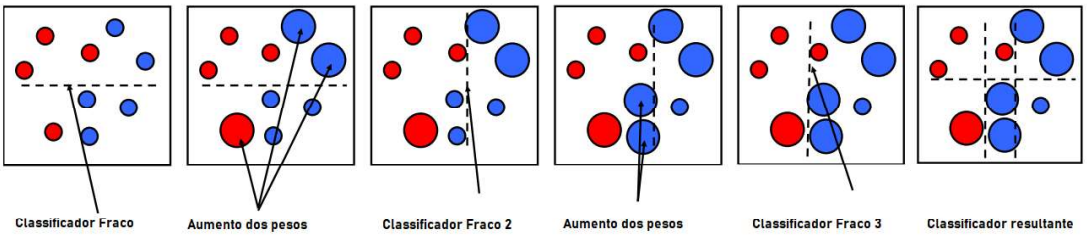
Um classificador forte (H(x)) é dado então pela Equação 1

$$H(x) = sign[\sum_{j=0}^{m-1} \alpha_j h_j(x)] \tag{1}$$

onde m é o numero de classificadores fracos, α_j é o peso do classificador j, h_j é o j-ésimo classificador fraco e x, no caso do Viola-Jones, é o resultado da soma do Atributo Haar sobre uma área da imagem. Cada classificador fraco retorna um valor positivo (geralmente +1) quando x é classificado em uma classe e um valor negativo (geralmente -1) quando é classificado em outra classe. A função *sign*, também chamada de função sinal, resulta -1 para valores negativos, 1 para valores positivos.

A Figura 6 demonstra um esquemático do funcionamento do *AdaBoost*: após cada classificador fraco escolhido, os pesos dos círculos classificados de forma errada são aumentados (isto é representado graficamente pelo aumento dos círculos). O resultado do classificador final é o sinal (+ ou -) da soma ponderada dos resultados dos classificadores fracos.

Figura 6 – Ilustração do funcionamento de *boosting*

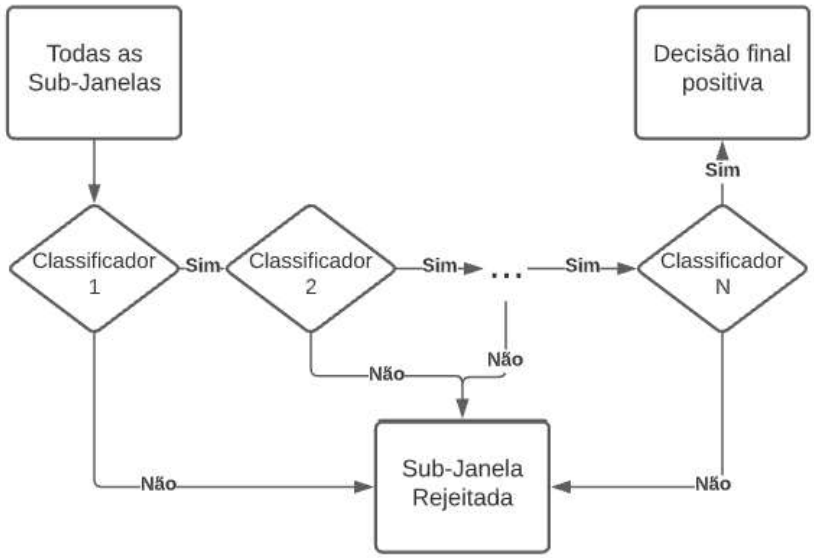


Fonte: Adaptado de Szeliski (2011).

2.2.4 Classificadores em cascata

Um classificador em cascata é um classificador de múltiplas etapas, onde um resultado positivo da etapa anterior inicia a próxima etapa, e um resultado negativo rejeita a sub-janela observada, como mostra a Figura 7. Cada uma dessas etapas consiste em um classificador forte gerado pelo *AdaBoost*, sendo que a cada etapa o classificador utilizado é formado de mais classificadores fracos e, portanto, requer mais processamento ao mesmo tempo que é mais preciso. A vantagem desse método é que, se ajustado para rejeitar o maior número possível de falsos positivos, mantendo um número aceitavel de falsos negativos, no início e ficando gradativamente mais restrito, uma quantidade grande de janelas já é eliminada pelos classificadores menos custosos e é possível manter uma velocidade rápida com uma precisão alta.

Figura 7 – Classificador em cascata com N etapas



Fonte: Adaptado de Viola e Jones (2001)

Para o treinamento desse classificador em cascata é utilizado um método prático onde são escolhidas as taxas mínimas de detecção e falso positivos de cada estágio e são adicionados classificadores fracos até que a taxa desejada seja atingida.

2.3 Rede Neural

De forma geral, uma rede neural é uma máquina modelada de modo a emular a forma que o cérebro realiza uma tarefa ou função de interesse. O cérebro é considerado um computador altamente complexo, capaz de realizar tarefas como reconhecimento de padrões (como

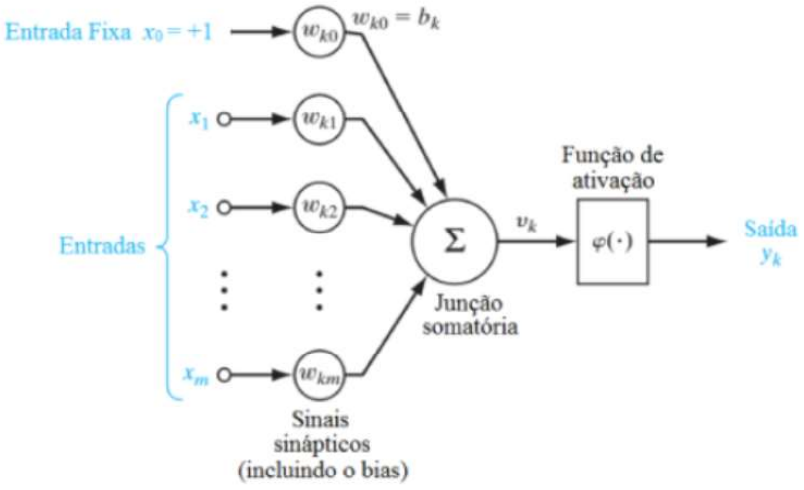
um rosto familiar) em aproximadamente 100-200 ms. Para tal, ele é capaz de arranjar seus componentes estruturais, denominados neurônios, para realizar o processamento de informações de forma muito rápida. De maneira análoga ao cérebro, uma rede neural artificial consiste em vários neurônios em camadas que podem ser interconectados de diversas maneiras. As configurações dessas redes geralmente são obtidas empiricamente, após a realização de testes de desempenho (HAYKIN, 2009).

2.3.1 Neurônio

Um neurônio é a unidade básica de processamento de uma rede neural. Como mostra a Figura 8, o neurônio artificial se comporta de forma semelhante ao natural, sendo que recebe sinais de entrada e gera um sinal de saída que passa então para outro neurônio ou como saída da rede. Os sinais de entrada representados por "x" são multiplicados por pesos "w" resultando em valores ponderados para cada entrada. Esses valores são então somados juntamente com o *bias* "b" e o resultado, representado por "v", passa então por uma função de ativação resultando no sinal de saída do neurônio. A função de ativação é utilizada para tratar a saída, em muitos casos isso se dá achatando (limitar) a amplitude do sinal de saída para uma faixa limitada de valores, como o intervalo -1 e 1, por exemplo.

O *bias*, um dos pesos de entrada de um neurônio, é um valor fixo que serve para ajudar o modelo utilizado a ter um *fit* melhor com os dados de treino. O processo de aprendizagem ocorre por meio de ajustes dos pesos sinápticos da rede até que se atinja (ou se aproxime) o objetivo desejado (GOEDERT; FILHO; BLANCO, 2017).

Figura 8 – Representação de um neurônio

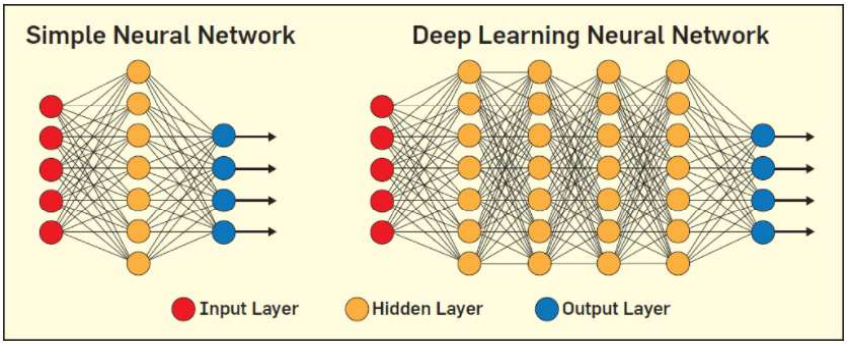


Fonte: Adaptado de Haykin (2009)

2.3.2 *Deep Learning*

Na última década, avanços em *deep learning* transformaram a comunidade de Inteligência Artificial (EDWARDS, 2018). Utilizando métodos de aprendizado como *backpropagation*, *deep learning* permite que modelos compostos de múltiplas camadas de processamento aprendam a representar dados com múltiplos níveis de abstração, diferentemente de redes neurais simples (Figura 9)(LECUN; BENGIO; HINTON, 2015).

Figura 9 – Comparação entre uma rede neural simples e uma rede neural profunda



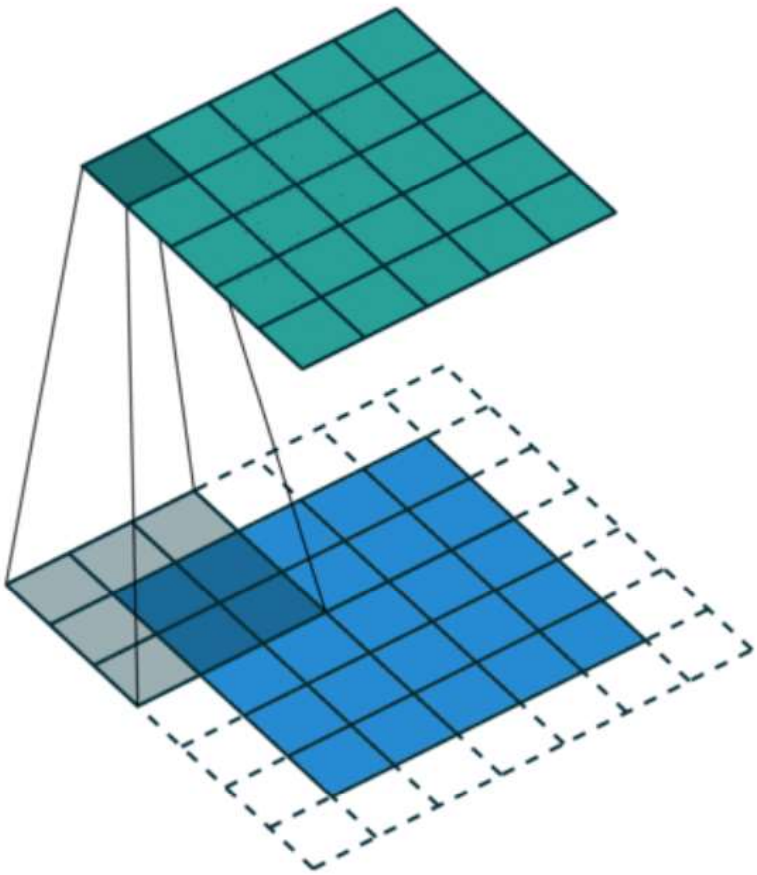
Fonte: Edwards (2018).

2.3.3 Rede Neural Convolucional (RNC)

Redes Neurais Convolucionais são redes neurais com uso primariamente na área de reconhecimento de imagem que, por meio da utilização de camadas convolucionais e camadas de pooling, reduzem de forma considerável a demanda de processamento". Após essas camadas iniciais, camadas totalmente conectadas são colocadas para gerar a classificação (O'SHEA; NASH, 2015).

A camada de convolução tem o objetivo primário de extrair características da imagem. Utilizando filtros treinados que realizam a operação de convolução em cima da imagem original, usando o método de janela deslizante, semelhante ao mencionado na Seção 2.2.1, gerando o chamado mapa de características. O mapa de características é controlado pelos atributos de: (*i*) profundidade, que depende da quantidade de filtros utilizados, (*ii*) passo, que é referente a quanto se move a janela quando é deslizada e (*iii*) *zero-padding*, que é a colocação ou não de zeros no exterior da matriz (ou tensor) para melhorar a extração de informações nas bordas da imagem (KARN, 2016). A Figura 10 mostra a convolução da imagem de entrada em azul: o filtro 3×3 sombreado na entrada realiza a convolução da área sombreada gerando uma célula na matriz de saída. A parte tracejada da matriz de entrada representa o *padding* que são células de valor zero adicionados ao redor da matriz.

Figura 10 – Funcionamento de uma camada de *convolução* com filtro 3×3 e passo 1 e *padding* 1

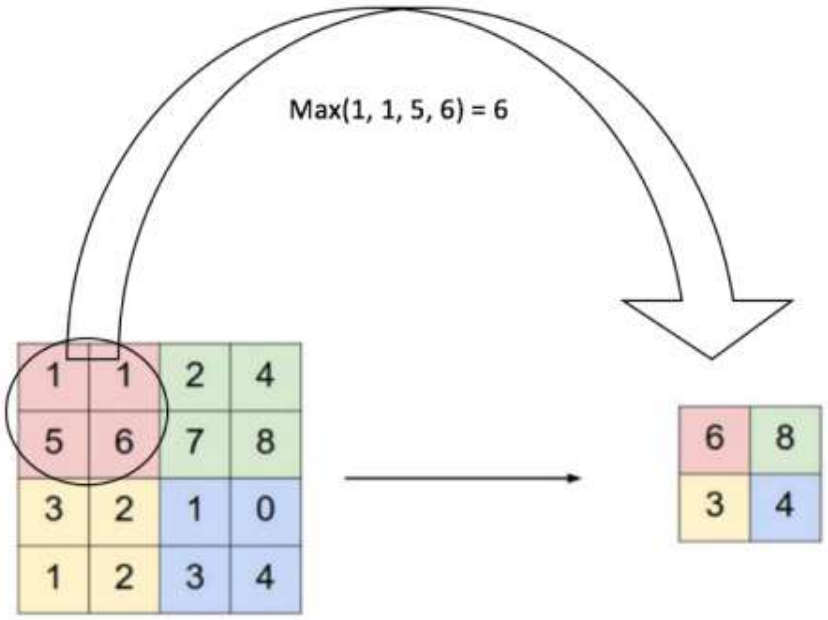


Fonte: Kamali (2016).

Já as camadas de *pooling*, também chamadas de *subsampling*, servem para reduzir a dimensão do mapa de características enquanto retém as informações mais relevantes. Essas camadas também ajudam a rede a se tornar mais robusta contra ruídos (KARN, 2016). A Figura 11 demonstra o funcionamento de uma camada de *max pooling* com passo (*stride*) 2, cada cor indica um passo do filtro 2×2 e o valor resultante da nova matriz é o valor máximo dentre as células.

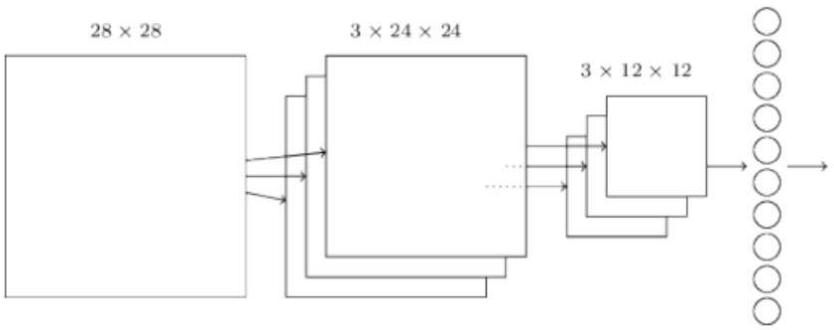
A Figura 12 apresenta uma arquitetura simplificada de uma RNC. Pode-se observar uma imagem de entrada 28×28 que passa pela convolução usando 3 filtros de 5×5 na primeira camada com passo 1 transformando em 3 imagens 24×24. Na segunda é aplicada uma camada de *pooling* 2×2 reduzindo e simplificando para 3 imagens 12×12 e por fim uma camada totalmente conectada que resulta na saída.

Figura 11 – Funcionamento de uma camada de *pooling* com filtro 2×2 e passo 2



Fonte: Karn (2016).

Figura 12 – Arquitetura simplificada de uma RNC



Fonte: Nielsen (2018).

2.4 YOLO

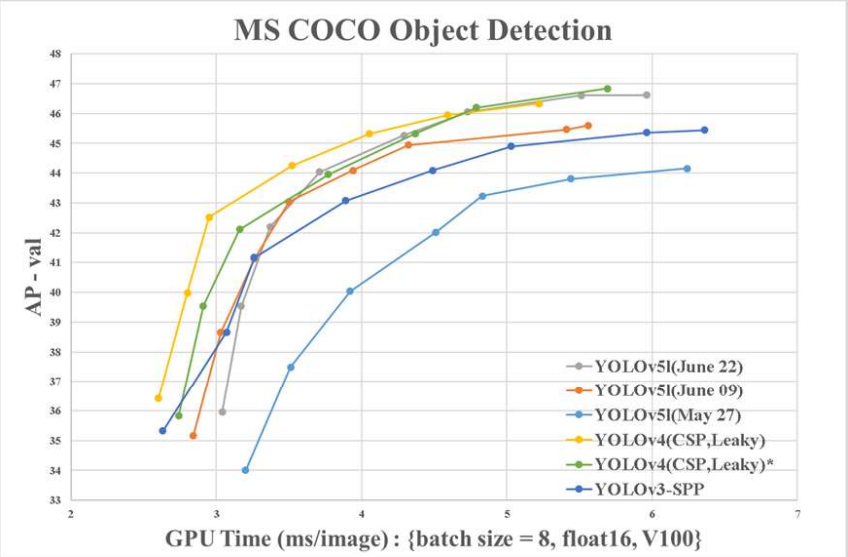
A detecção de objetos é um problema clássico na visão computacional e apesar disso os métodos e soluções para localizar e classificar um objeto em uma imagem normalmente tratavam esse problema em duas etapas: (*i*) Extrair diferentes áreas da imagem usando janelas deslizantes de diferentes tamanhos, (*ii*) Aplicar alguma abordagem de classificação para determinar qual classe os objetos encontrados pertencem. Esse tipo de abordagem sofre com dificuldade de otimização das etapas em termos de velocidade e requer muito poder de processamento.

Desenvolvido por Joseph Redmon e colaboradores, o algoritmo YOLO (*You Only Look*

Once) foi introduzido em 2015 no campo de detecção de objetos com um sistema de detecção que utiliza apenas uma rede neural para realizar todas as etapas essenciais da detecção. Esse algoritmo foi um dos primeiros a transformar o problema de detecção para um único problema de regressão direto das imagens para localização e nível de confiança de classes. Isso trouxe uma das maiores vantagens da YOLO em relação a outros algoritmos de detecção de imagem, que é sua velocidade de predição que permite que seja utilizada em sistemas em tempo real além de possuir boa precisão (REDMON et al., 2016).

Durante os últimos anos a YOLO sofreu diversas modificações que melhoraram seu desempenho, sendo as versões mais recentes a YOLOv4 e a YOLOv5. Apesar dos nomes, ambas as versões são derivadas da YOLOv3 e são projetos paralelos. Enquanto a YOLOv4 continua usando a arquitetura Darknet e a linguagem das versões anteriores, a YOLOv5 foi totalmente re-escrita em Python usando o PyTorch em 2020. Como pode ser observado na Figura 13, que apresenta um comparativo entre precisão (*Ap*) e velocidade (*ms/Image*) de algumas versões da YOLO, a diferença de desempenho entre ambas não é muito significativa e, portanto, a YOLOv5 foi usada por questão de facilidade de uso.

Figura 13 – Comparação entre diferentes versões da YOLO



Fonte: Nelson (2020).

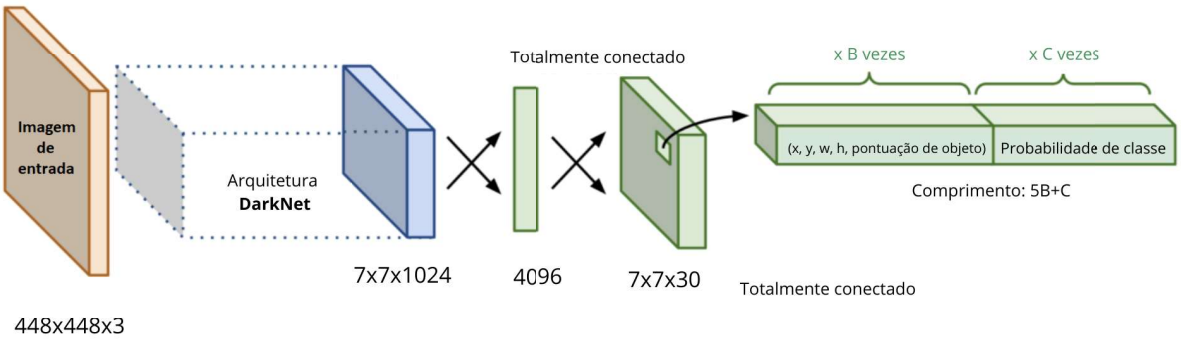
2.4.1 Princípios

A principal ideia desse método é dividir a imagem de entrada em uma grade de $S \times S$ células e se o centro de um objeto estiver dentro de uma dessas é responsabilidade dessa célula detectar o objeto. Para cada uma dessas células é predito um número B de *bounding boxes* e a predição de qual é o objeto detectado, além de um nível de confiança de classe (entre 0 e 1) para a própria célula dentre as C classes de objetos a serem detectadas, como

demonstrado na Figura 15. Com isso, a rede é capaz de, por meio de convoluções, reduzir o volume de entrada para um tensor $S \times S \times (B \times 5 + C)$ (REDMON et al., 2016).

A Figura 14 traz a arquitetura da YOLO. Ela possui 24 camadas convolucionais, no que os autores chamam de arquitetura *Darknet*, seguidas de duas camadas totalmente conectadas. Essas reduzem a imagem inicial $448 \times 448 \times 3$ para um tensor de tamanho $7 \times 7 \times 30$, onde 7×7 é o tamanho da grade escolhida e 30 é composto pela quantidade de *bounding boxes* gerada por cada célula (2) multiplicada por 5 que é a quantidade de variáveis que cada *bounding box* prediz (posição em x, posição em y, largura, comprimento e confiança da *bounding box*), somada da quantidade de classes da figura de exemplo (20).

Figura 14 – Arquitetura simplificada da YOLO

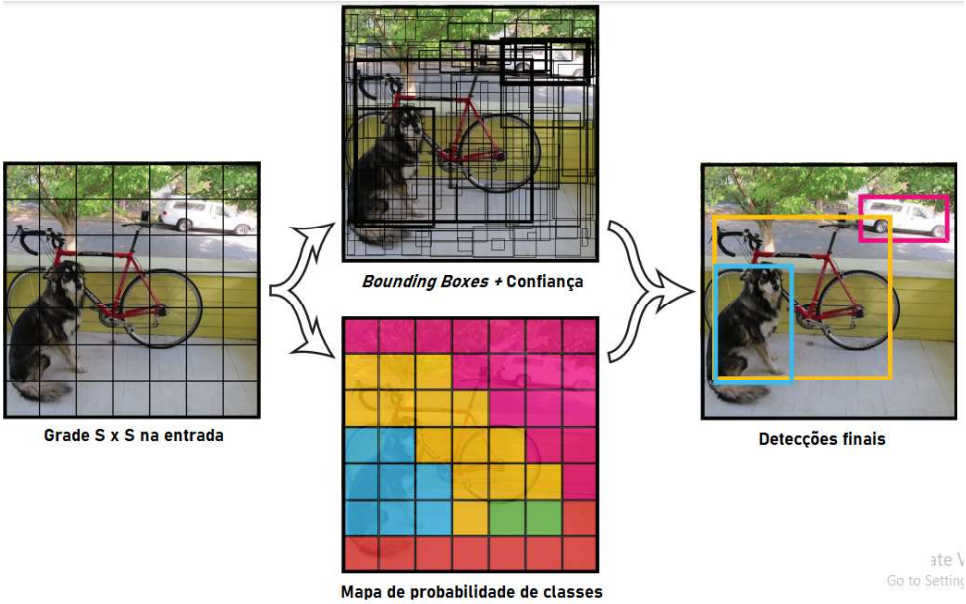


Fonte: Thuan (2021).

Na Figura 15 podem ser vistas que são criadas muitas *bounding boxes*. As melhores são selecionadas através dos chamados *non-maximum suppression* e *intersection over union* (IOU), que servem para retirar *bounding boxes* de confiança baixa e repetidas de acordo com limiares escolhidos.

Neste trabalho foi utilizado a versão da YOLO conhecida como YOLOv5 assim, serão citadas algumas mudanças que o método sofreu ao longo de sua evolução, vale ressaltar que já existem trabalhos que descrevem com maior profundidade as diferenças da primeira versão da YOLO até a YOLOv5 como Thuan (2021). De forma geral as alterações foram na arquitetura da rede, incluindo: *(i)* a utilização de CSPNet (*Cross Stage Partial Networks*) como *Backbone*; *(ii)* PANet (*Path Aggregation Network*) como *Neck* e *(iii)* como *Head*

Figura 15 – Estrutura de funcionamento da YOLO



Fonte: Redmon et al. (2016).

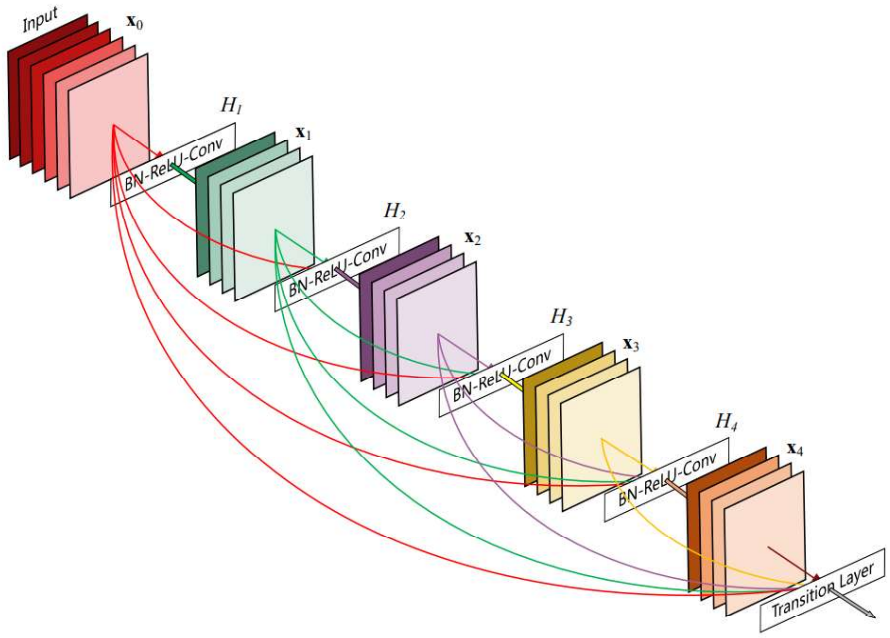
é utilizado a mesma metodologia da YOLOv3 para predição a qual, diferentemente da YOLO, realiza predições sem as camadas totalmente conectadas, utilizando a chamada *anchor box architecture*, e em diferentes resoluções melhorando o problema da rede com relações a pequenos objetos.

2.4.2 *Cross Stage Partial Networks* (CSPNet)

A CSPNet é um *backbone* apresentado por Wang et al. (2019). Ela se baseia na *Densely Connected Convolutional Networks* (*DenseNet*) apresentada em Huang et al. (2017), porém Wang et al. (2019) leva seu conceito um passo adiante. A premissa da *DenseNet* é que redes neurais convolucionais podem ser mais profundas, precisas e eficientes no treinamento se possuírem conexões mais curtas de camadas próximas da entrada com as camadas próximas à saída. Assim, a *DenseNet* utiliza a ideia de concatenação de mapas de *features* de mesmo tamanho para não apenas melhorar sua precisão como também para diminuir a necessidade de poder computacional por meio da diminuição da quantidade de parâmetros (filtros) entre camadas (HUANG et al., 2017).

A Figura 16 apresenta um exemplo do *layout* de concatenação presente no chamado *Dense Block*, nele os mapas de *features* de cada etapa anterior são concatenados aos mapas resultantes das novas entradas, como cada etapa possui 4 filtros é dito que esse bloco tem a taxa de crescimento igual a 4.

Figura 16 – Exemplo do layout de um *Dense Block*

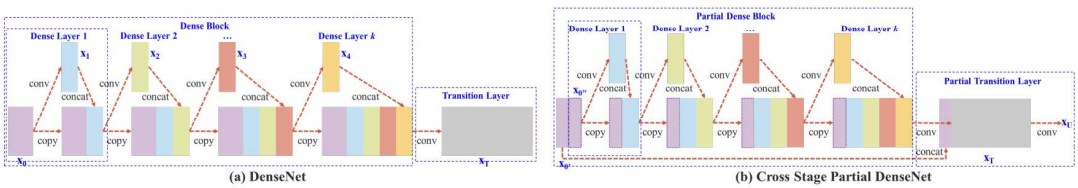


Fonte: Huang et al. (2017).

Uma *DenseNet* é formada por vários desses blocos que são separados por camadas de *pooling* já que não é possível concatenar mapas de tamanhos diferentes. Um aspecto contraintuitivo desse tipo de rede é que ele é mais rápido apesar de sua concatenação de mapas, pois ele acaba necessitando de menos filtros comparado a redes com precisões semelhantes (HUANG et al., 2017).

De maneira semelhante, a ideia apresentada pela CSPNet utiliza um sistema semelhante de concatenação dos mapas de *features*, porém os mapas presentes na entrada de cada bloco são divididos e uma parte vai diretamente ser concatenada com a saída, enquanto a outra passa normalmente pelo bloco. Um dos efeitos dessa mudança é a redução ainda maior da necessidade de poder computacional. Uma comparação entre as duas ideias pode ser vista na Figura 17.

Figura 17 – Comparação entre os blocos da DenseNet e da CSPNet



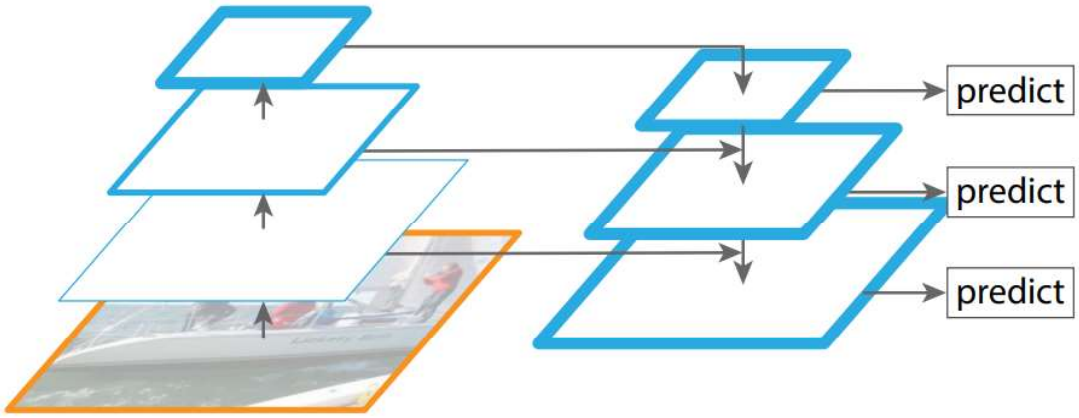
Fonte: Adaptado de Wang et al. (2019).

2.4.3 *Path Aggregation Network* (PANet)

A PANet é um *neck* e, como citado anteriormente, esse é utilizado para extrair *features* mais elaboradas do resultado obtido do *backbone*. O princípio utilizado pela PANet surgiu com as *Feature Pyramid Networks* (FPN). Essas utilizam as *features* dos estágios mais profundos do *bottom-up path*, que nada mais são do que as *features* geradas no final do *backbone*. A escala das *features* é reduzida ao longo da rede e quanto mais perto do fim elas são consideradas como as *features* mais fortes e mais semânticas. As *features* selecionadas sofrem, então, o processo reverso e, utilizando o método de *nearest neighbors*, passam pelo chamado *upsampling* no *top-down path*, que vai gerar *features* semânticas, porém espacialmente pobres o que é compensado pelas conexões laterais que mesclam os mapas de *features* entre as etapas de mesma escala do *bottom-up path* e do *top-down path* e também facilitam o treino. A predição é então realizada em diversas escalas ao longo do *top-down path* (LIN et al., 2017).

A Figura 18 mostra o *layout* básico de uma FPN, na esquerda a "pirâmide" feita pelo backbone, o chamado *bottom-up path*, e na direita pode ser observada a "pirâmide" gerada no *top-down path*. As setas entre elas indicam os pontos de entrada e conexão lateral entre elas.

Figura 18 – Estrutura básica de uma FPN

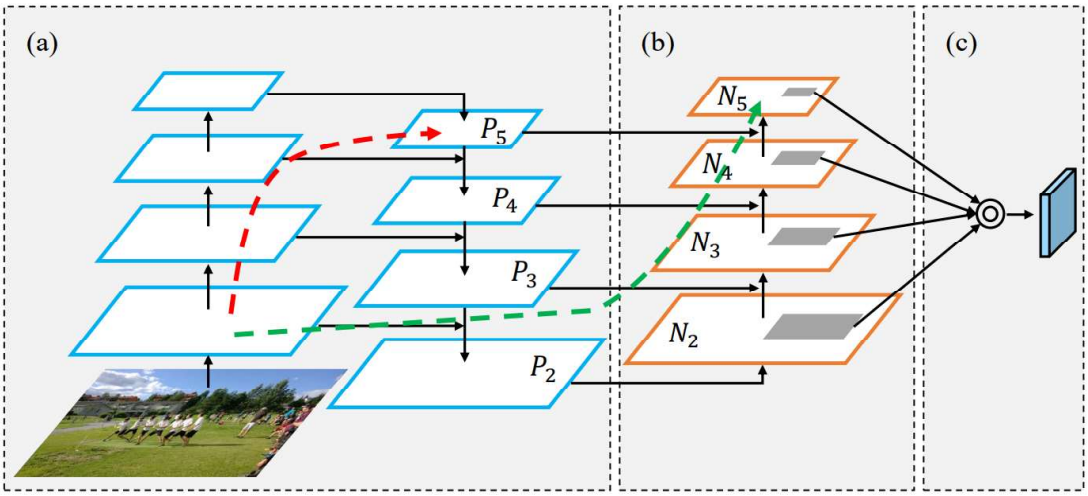


Fonte: Lin et al. (2017)

Uma PANet usa, então, uma arquitetura similar, porém, como pode ser visto na Figura 19, ela adiciona um segundo *bottom-up path* para melhorar a precisão da localização de *features* (b), ela também, contrário a uma FPN normal que realiza as predições em várias escalas diferentes, utiliza *pooling* adaptativo para combinar as *features* de todas as etapas antes das predições (c). Além disso, a rede tem um caminho lateral mais curto, representado pela seta verde, de cerca de 10 camadas em comparação com o caminho normal da informação que pode passar por mais de 100 camadas, representado pela seta vermelha que melhora

ainda mais a propagação de *features* de baixo nível e melhora a capacidade de localização da rede.

Figura 19 – Estrutura da PANet



Fonte: Adaptado de Liu et al. (2018)

3 PROPOSTA

O avanço da tecnologia é muito rápido e técnicas que a dez anos eram estado da arte muitas vezes já se tornaram obsoletas. Nesse projeto foi realizado um estudo de caso de comparação entre dois algoritmos utilizados para detecção facial a fim de verificar quantitativamente a evolução das técnicas utilizadas para este fim. Para estimar essa diferença, foram feitas predições utilizando cada algoritmo em dois conjuntos de dados destinados à tarefa de detecção facial.

Neste capítulo serão apresentadas a metodologia e as ferramentas utilizadas nos experimentos realizados. Desta maneira, o capítulo começa com a descrição dos recursos utilizados, como dos conjuntos de dados, bibliotecas, softwares e recursos físicos. Ademais, são expostas as métricas utilizadas para a análise de desempenho dos algoritmos. Os repositórios para as bibliotecas e códigos utilizados podem ser emcontrados no Anexo .1.

3.1 Conjuntos de dados

Os conjuntos de dados escolhidos neste projeto foram: *Wider Face* e *Face Detection Data Set and Benchmark* (FDDB). Ambos são conjuntos de imagens públicas já consolidados na comunidade de detecção facial e apresentam grande quantidade de imagens de tamanhos variados e com faces de diversos graus de escala e dificuldade de detecção como mostram as Tabelas 1 e 2. Na tabela 2 são apresentados os dados do conjunto completo *Wider Face* e a parte da validação e de treino, que foram os conjuntos utilizados nos experimentos.

Tabela 1 – Comparação da altura das faces presentes em pixels

Conjuntos de Dados	**10-50 px**	**50-300 px**	**>=300 px**
FDDB	8%	86%	6%
Wider Face	50%	43%	7%

Fonte: Adaptado de (YANG et al., 2016)

3.1.1 *Wider Face*

O conjuntos de dados *Wider Face* foi elaborado por Yang et al. (2016) para servir de *benchmark* e introduzir um conjunto de dados efetivo de treinamento de redes neurais para detecção facial. A proposta dos autores era de que existia uma distância entre a

Tabela 2 – Comparação da quantidade de imagens e faces

Conjuntos de Dados	**imagens**	**faces**	**média faces/imagem**
FDDB	2.845	5.171	1, 82
Wider Face	32.203	393.703	12, 23
Wider Face Validação	3.226	39.708	12, 31
Wider Face Treino	12.880	159.424	12, 37

Fonte: Produção do próprio autor.

capacidade dos sistemas de detecção e as necessidades existentes. Assim, foi introduzido o conjunto de dados *Wider Face* apresentando, na época de seu lançamento, um conjunto dez vezes maior do que qualquer outro existente, contendo 32.203 imagens e 393.703 faces com anotações que incluem não apenas a localização, mas também pose e oclusão. Como pode ser visto na Figura 20, o conjunto de dados possui muita variação nas faces presentes nas imagens e por isso é considerado possuir um alto grau de dificuldade para a detecção. As marcações em verde representam as anotações dos rostos nas imagens.

Figura 20 – Exemplos de imagens do *Wider Face*

Fonte: Adaptado de Yang et al. (2016).

3.1.2 *Face Detection Data Set and Benchmark* (FDDB)

O conjunto FDDB foi desenvolvido por Jain e Learned-Miller (2010) com o objetivo de servir para o estudo do problema de detecção facial, contendo 2485 imagens com um total de 5171 faces. A FDDB conta com imagens coloridas e em escala de cinza com variações de pose e oclusão. Ainda assim é um conjunto de dados que, comparado ao *Wider Face*, possui uma dificuldade menor para detecção das faces presentes pois suas imagens apresentam, em média, faces maiores e em posição frontal.

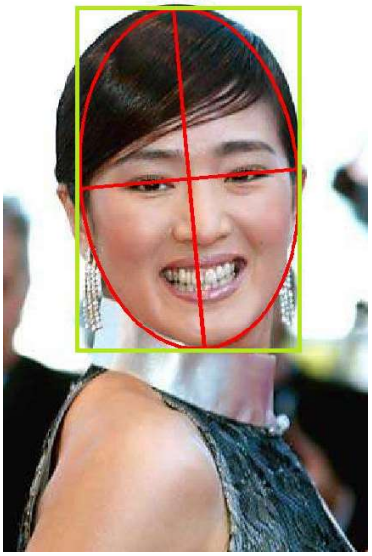
Diferentemente de outros conjuntos de dados, o FDDB apresenta a delimitação das faces usando elipses como pode ser visto na Figura 21, com as marcações em vermelho representando as anotações das faces na imagem.

Figura 21 – Exemplos de imagens do FDDB

Fonte: Adaptado de Jain e Learned-Miller (2010).

Para a utilização no projeto se fez necessária a conversão do formato da anotação para uma anotação retangular. Para tal, foi feito um script em Python para converter as anotações utilizando os pontos mais extremos da elipse para demarcar o retângulo. A Figura 22 exemplifica o posicionamento da nova anotação, representada pelo retângulo verde.

Figura 22 – Exemplos do posicionamento da anotação após conversão



Fonte: Adaptado de Jain e Learned-Miller (2010).

3.2 Recursos Computacionais

3.2.1 Recursos de *Software*

O projeto foi realizado totalmente em Python, uma linguagem interpretada de alto nível e com grande diversidade de pacotes. Sua facilidade de escrita e grande comunidade na área de visão computacional e redes neurais a tornam ideal para abordagem desse tipo de

problema. Em conjunto com isso, a capacidade de criação de ambientes de desenvolvimento virtual faz com que a criação de múltiplos projetos em um computador não gere problemas com pacotes de forma global.

O projeto da YOLOv5 é desenvolvido usando o PyTorch, um *framework open-source* de tensores com foco em usabilidade e eficiência usado primariamente em problemas de *Deep Learning* desenvolvida pelo time de pesquisa de inteligência artificial do Facebook (PASZKE et al., 2019). É uma das mais utilizadas bibliotecas de aprendizado de máquina, rivalizando com o TensorFlow. Por outro lado, o projeto da Viola-Jones foi realizado usando a biblioteca OpenCV, que é um pacote de funções de programação focadas em visão computacional em tempo real. Originalmente desenvolvida pela Intel, é uma biblioteca *open source* e multiplataforma.

3.2.2 Recursos de *Hardware*

A máquina utilizada nos experimentos foi disponibilizada pelo autor e possui a seguinte configuração: (*i*) sistema operacional Windows 10; (*ii*) processador AMD Ryzen 5 3600x, 3.80GHz com 6 núcleos físicos; (*iii*) memória RAM de 16 GB; (*iv*) unidade de armazenamento de 512GB (M2 NVMe); (*v*) placa de vídeo AMD PowerColor Red Devil RX 5700XT, com 8 GB de memória dedicada. Também foi utilizado para os testes de desempenho uma máquina com disponibilidade de núcleos CUDA com as seguintes especificações: (*i*) sistema operacional Windows 10; (*ii*) processador AMD Ryzen 5 5600x, 3.80GHz com 6 núcleos físicos; (*iii*) memória RAM de 32 GB; (*iv*) unidade de armazenamento de 1TB (M2 NVMe); (*v*) placa de vídeo NVIDIA GeForce GTX 1650 SUPER, com 4 GB de memória dedicada e 869 núcleos CUDA.

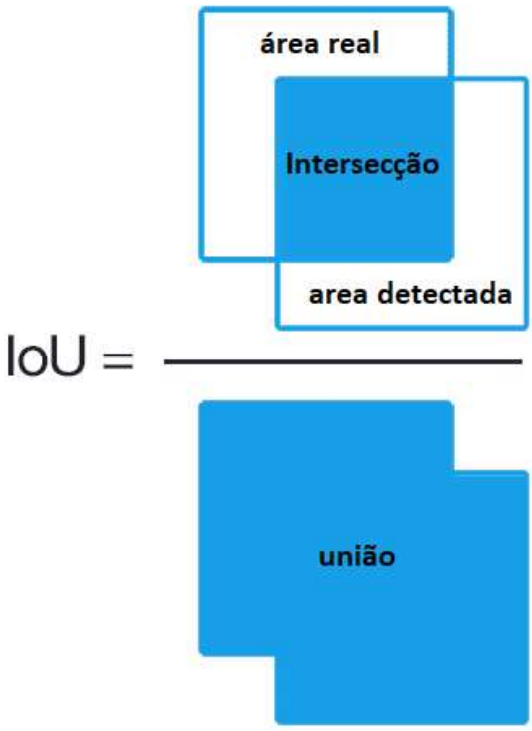
3.3 Métricas

Uma das métricas mais comuns e populares para verificar se áreas detectadas correspondem a um objeto é a Intersecção sobre União (IoU). Essa métrica se difundiu com mais força no contexto de visão computacional por causa do desafio Pascal VOC (EVERINGHAM et al., 2010). O funcionamento dela é computar a razão entre a intersecção das áreas (área detectada e área real do objeto) pela união dessas mesmas áreas como mostra a Equação 2. O resultado é um valor entre 0 e 1 que é comparado com um valor mínimo (geralmente 0,5) para verificar se as áreas correspondem ou não a um mesmo objeto. A Figura 23 exemplifica o cálculo de IoU, representado pela divisão das áreas preenchidas de azul.

$$IoU = \frac{area(a1 \cap a2)}{area(a1 \cup a2)} \tag{2}$$

onde a1 é a área detectada e a2 é a área real do objeto.

Figura 23 – Representação visual do calculo de IoU.



Fonte: Produção própria do Autor.

Devido ao método Viola-Jones não apresentar um valor de confiança em suas predições, as métricas selecionadas para a comparação dos métodos neste trabalho foram a precisão (P), o recall (R), acurácia (Ac) e a medida F1. A partir das detecções feitas sobre o conjunto imagens, pode-se definir a precisão como a relação entre a quantidade de detecções corretas e o número total de detecções, por outro lado, o recall é calculado como a razão entre detecções corretas e o número total de objetos na imagem. Ambas as métricas podem ser encontradas pelas equações 3 e 4, respectivamente. A acurácia pode ser definida como a relação da quantidade de detecções corretas pelo número total de objetos somado das detecções erradas (Equação 5). Por fim, a medida F1 é a média harmônica entre a precisão e o recall (Equação 6).

$$P = \frac{VP}{VP + FP} \tag{3}$$

$$R = \frac{VP}{VP + FN} \tag{4}$$

$$Ac = \frac{VP}{VP + FP + FN} \tag{5}$$

$$F1 = \frac{2 \times P \times R}{R + P} \tag{6}$$

onde P é a precisão, R o recall, os verdadeiros positivos (VP) são as detecções corretas, falsos positivos (FP) são as detecções incorretas e falsos negativos (FN) são os objetos não detectados.

3.4 Treinamento da Yolov5

A YOLO foi inicialmente pensada como uma rede neural para detecção de objetos e seus modelos pré-treinados são treinados no conjunto de imagens COCO, que contém 80 classes diferentes para detecção. Porém, neste projeto, o interesse é de apenas detectar faces, logo, foi necessário treinar uma rede especificamente para esta tarefa.

3.4.1 Conjunto de dados de treinamento

Para o treino da YOLO foi escolhido o conjunto de treinamento fornecido pela *Wider Face* contendo 12.880 imagens e mais de 150.000 faces rotuladas.

As marcações das faces tiveram que ser normalizadas pelo tamanho da imagem para valores no intervalo de 0 a 1. Como o conjunto da *Wider Face* apresenta imagens de tamanhos variados, foi necessário verificar cada uma para normalizar os valores. Por fim, antes de cada marcação é necessário também indicar a classe a qual ela pertence, como foi utilizado apenas 1 classe para todas as marcações foi colocado um 0 antes de todas as marcações. A Figura 24 mostra um exemplo de arquivo de marcação alterado para melhor visualização.

Desse conjunto foram separados 20% das imagens para serem usadas na validação do treinamento da rede.

A YOLOv5 acessa esse conjunto de dados e usa elas de entrada para o treinamento por meio de um arquivo *.yaml* que contém as informações básicas do conjunto utilizado (Figura 25). Esse arquivo segue a seguinte estrutura:

Figura 24 – Exemplo de valores de marcação após ajustes

#classe	x1	y1	comprimento	altura
0	0.0449	0.3865	0.0332	0.0615
0	0.0088	0.358	0.0195	0.063
0	0.1357	0.3397	0.0332	0.0586
0	0.2144	0.2987	0.0479	0.082
0	0.4014	0.172	0.0977	0.1567
0	0.374	0.3155	0.041	0.0747
0	0.4966	0.3843	0.0283	0.0542
0	0.4111	0.3602	0.0156	0.0351
0	0.6016	0.1852	0.0879	0.1508

Fonte: Produção própria do Autor.

1. path: <caminho para o diretório contendo todas as imagens>
2. train: <caminho relativo a *path* para o diretório de imagens de treino>
3. val: <caminho relativo a *path* para o diretório de imagens de validação>
4. nc: <número de classes>
5. names: <nome das classes>

Figura 25 – Arquivo *yaml* utilizado

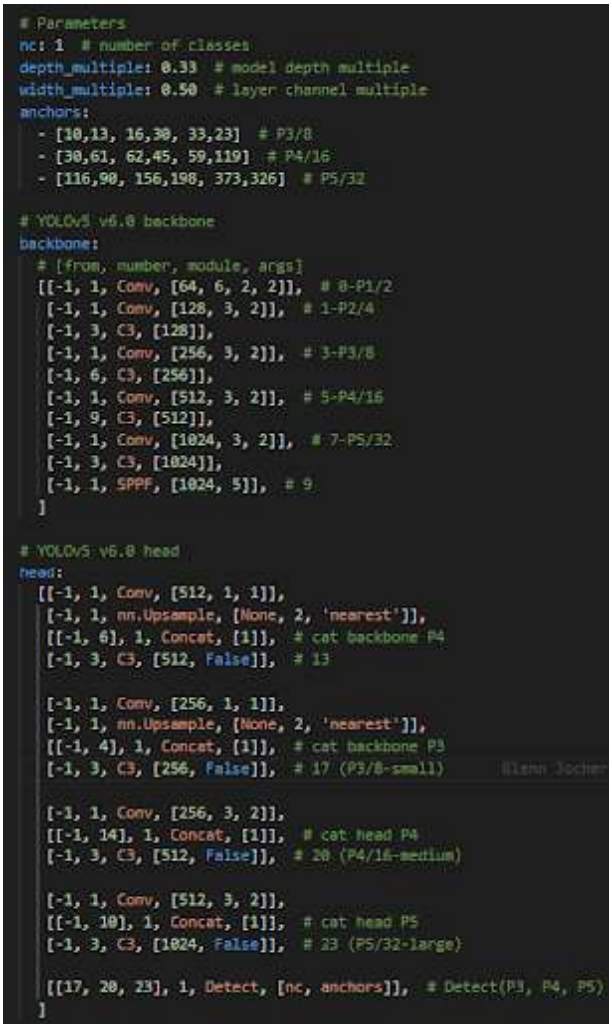
```
1  path: C:\Users\T-GAMER\Desktop\YOLOV5\data\
2  train: images\
3  val: validation\images\
4
5  nc: 1
6  names: ['Face']
```

Fonte: Produção própria do Autor.

3.4.2 Arquitetura

A YOLOv5 oferece alguns modelos de arquitetura pré-prontos, a utilizada foi a *yolov5s.yaml* (Figura 25), essa é uma das menores em questão de complexidade porém é umas das mais eficientes para realizar predições. Como o objetivo do projeto é analisar o desempenho da arquitetura original, nenhuma modificação foi realizada na arquitetura a não ser modificar o número de classes, todavia, como o programa lê a arquitetura de um arquivo *yaml*, esses modelos são facilmente customizáveis para diferentes problemas de detecção.

Figura 26 – Arquivo de arquitetura *yolov5s.yaml*



```yaml
# Parameters
nc: 1  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 v6.0 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Conv, [64, 6, 2, 2]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, C3, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 6, C3, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, C3, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 3, C3, [1024]],
   [-1, 1, SPPF, [1024, 5]],  # 9
  ]

# YOLOv5 v6.0 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, C3, [512, False]],  # 13

   [-1, 1, Conv, [256, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 4], 1, Concat, [1]],  # cat backbone P3
   [-1, 3, C3, [256, False]],  # 17 (P3/8-small)

   [-1, 1, Conv, [256, 3, 2]],
   [[-1, 14], 1, Concat, [1]],  # cat head P4
   [-1, 3, C3, [512, False]],  # 20 (P4/16-medium)

   [-1, 1, Conv, [512, 3, 2]],
   [[-1, 10], 1, Concat, [1]],  # cat head P5
   [-1, 3, C3, [1024, False]],  # 23 (P5/32-large)

   [[17, 20, 23], 1, Detect, [nc, anchors]],  # Detect(P3, P4, P5)
  ]
```

Fonte: Produção própria do Autor.

3.4.3 Treinamento do Modelo

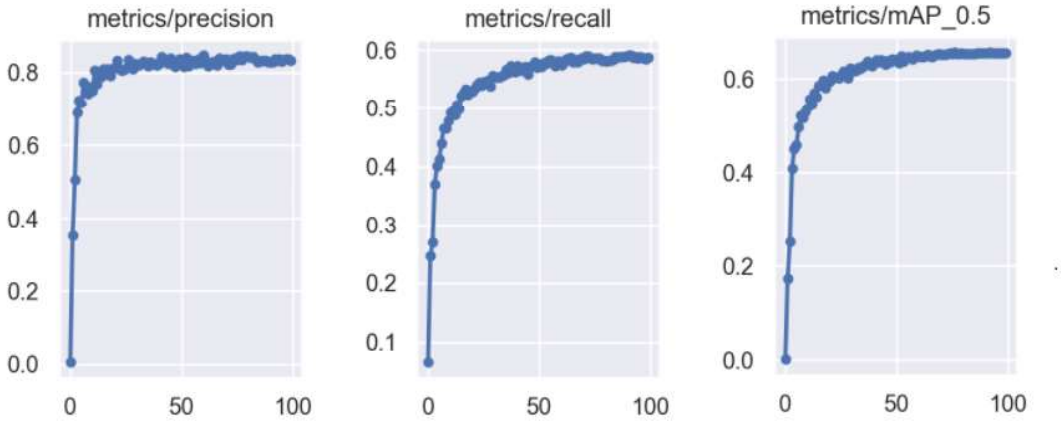
O treinamento foi realizado utilizando em sua maioria as configurações padrões já existentes (Tabela 3). O tempo de treinamento foi longo devido a 3 motivos: Devido ao alto número de imagens; ao tamanho de entrada da imagem que foi colocado em 640 para evitar perder muitos detalhes pois o conjunto *Wider Face* apresenta muitas faces em escala pequena; e a indisponibilidade de núcleos CUDA no equipamento utilizado pois a YOLOv5 é totalmente otimizada para uso dos mesmos. Por esses motivos o tempo de treinamento de cada época foi em média de 1:30h e a de validação 25 minutos. Após 100 épocas, que é o mínimo recomendado, verificou-se ganhos marginais na continuação do treinamento e por isso foi decidido usar o melhor modelo encontrado até então. A Figura 27 apresenta os resultados do modelo treinado após cada época.

Tabela 3 – Hiperparâmetros de treino da YOLO

Parâmetro	**Valor**
Épocas	100
Tamanho de batch	16
Tamanho de imagem	640
Taxa de aprendizado inicial	$0,01$
Taxa de aprendizado final	$0,001$
Momento	$0,973$
IoU de treino	$0,2$

Fonte: Produção do próprio autor

Figura 27 – Resultados do treino da YOLO



Fonte: Produção própria do Autor.

4 RESULTADOS

4.1 Resultados

O experimento realizado compara a YOLO e a Viola-Jones fazendo a predição de faces para o conjunto de validação do *Wider Face* e para o conjunto de imagens completo do FDDB. Basicamente é analisada a diferença de desempenho entre os dois métodos considerando as métricas estabelecidas. Também é avaliada a eficiência de cada método, neste é analisado o tempo médio por imagem que os métodos levam para inferir a existência de faces nas imagens.

4.1.1 Comparação de desempenho

Os resultados dos experimentos realizados no conjunto de dados FDDB são apresentados na Tabela 4. Os resultados dos experimentos no conjunto de validação da *Wider Face* são apresentados na Tabela 5. Neles podem ser observados o comportamento da saída para casos distintos: (*i*) Predição com Viola-Jones e (*ii*) predição com YOLO utilizando diferentes níveis de confiança, mais especificamente com os limiares mínimos de 0,2 a 0,9, com aumento com passos de 0,1.

Tabela 4 – Resultados no conjunto *FDDB*

Método	**Precisão(%)**	**Recall(%)**	**Acurácia(%)**	**F1(%)**
Viola-Jones	88,33	70,23	64,27	78,25
Yolo confiança 0,2	82,60	**95,09**	79,22	88,40
Yolo confiança 0,3	86,41	94,73	82,45	90,38
Yolo confiança 0,4	89,28	94,15	84,59	91,65
Yolo confiança 0,5	91,69	92,66	**85,49**	**92,17**
Yolo confiança 0,6	94,01	88,93	84,16	91,40
Yolo confiança 0,7	96,31	73,12	71,12	83,12
Yolo confiança 0,8	98,62	37,36	37,17	54,19
Yolo confiança 0,9	**100**	0,001	0,001	0,003

Fonte: Produção do próprio autor

Primeiramente, se tratando apenas do Viola-Jones. Pode ser observado uma diferença muito grande entre as métricas das predições realizadas em cada conjunto de imagens. Enquanto na FDDB os resultados foram razoáveis, com 64,27% de acurácia e 78,25% na F1 (Figura 28), os resultados foram muito ruins na *Wider Face*, apresentando apenas 16% de acurácia. Isso se deve ao elevado grau de dificuldade do conjunto de dados *Wider Face*,

Tabela 5 – Resultados no conjunto *Wider Face*

Método	**Precisão(%)**	**Recall(%)**	**Acurácia(%)**	**F1(%)**
Viola-Jones	78, 91	17, 33	16, 56	28, 42
Yolo confiança 0,2	74, 67	**67, 15**	54, 69	70, 71
Yolo confiança 0,3	84, 14	63, 30	**56, 55**	**72, 25**
Yolo confiança 0,4	90, 17	58, 19	54, 71	70, 72
Yolo confiança 0,5	94, 43	51, 95	50, 41	67, 03
Yolo confiança 0,6	97, 56	42, 96	42, 21	59, 66
Yolo confiança 0,7	99, 16	30, 57	30, 49	46, 73
Yolo confiança 0,8	99, 80	14, 14	14, 13	24, 77
Yolo confiança 0,9	**100**	0, 02	0, 02	0, 04

Fonte: Produção do próprio autor

como pode ser observado na Figura 29. O método Viola-Jones sofre muito quando ocorre oclusão, mudanças de iluminação, poses diferentes e faces em escalas pequenas que são características muito presentes nas imagens desse conjunto.

Figura 28 – Detecções com Viola-Jones no conjunto *FDDB*

Fonte: Produção própria do Autor.

Por outro lado, analisando de forma geral os resultados, o método YOLO demonstra o poder do aprendizado presente em uma rede neural: em apenas 100 épocas a rede treinada foi capaz de aprender o suficiente para obter resultados razoavelmente satisfatórios no conjunto *Wider Face* (Figura 31) para, quando utilizada o valor de confiança $0,3$, obter uma acurácia de $56,55\%$ e um F1 de $72,25\%$. Como é explicado na literatura, o desempenho da YOLO cai a medida que o valor mínimo de confiança aceitável aumenta, com exceção da métrica precisão, a qual fica cada vez mais elevada, até o valor máximo de 100%. Apesar do aumento dessa confiança ser benéfico no quesito de precisão, o *trade off* com as demais métricas pode não valer a pena, dependendo da aplicação pretendida. Já quando utilizada no conjunto da FDDB os resultados apresentados foram bem melhores,

Figura 29 – Detecções com Viola-Jones no conjunto *Wider Face*

Fonte: Produção própria do Autor.

apresentando acurácia de 85,36% e por ser um conjunto de imagens mais "fáceis", sua perda de desempenho para valores maiores de confiança não sofre tanto, exceto para valores muito elevados de confiança (Figura 30). É interessante ressaltar também que para fins de comparação entre técnicas que utilizam redes neurais, é comum a utilização do valor de confiança como 0,5 e que sua alteração é uma forma de ajustar a rede para melhor se adaptar aos dados fornecidos, como poderia ser feito em aplicações reais.

Figura 30 – Detecções com YOLOv5 no conjunto FDDB

Fonte: Produção própria do Autor.

Em comparação direta, os resultados da YOLOv5 são superiores aos do Viola-Jones em quase todos os casos e, diferentemente dele, por não sofrer tanto com variações nas faces das imagens, se torna interessante também para aplicações onde é necessário uma acurácia

Figura 31 – Detecções com YOLOv5 no conjunto *Wider Face*

Fonte: Produção própria do Autor.

maior em situações variadas. Contudo, destaca-se que o fato da YOLOv5 ter sido treinada com parte das imagens do *Wider Face* colaborou para o seu melhor desempenho. Talvez a mesma conclusão poderia não ser alcançada se a YOLO fosse treinada com imagens menos desafiadoras. Também pode-se inferir que considerando a diferença de quase 20 anos entre os dois métodos, o método Viola-Jones apresenta uma acurácia satisfatória e uma precisão boa se comparada com a da YOLOv5, para casos em que a imagem apresenta a parte frontal da face, como mostrada na a Figura 32. Infelizmente sua dificuldade para detecção de faces em diferentes poses, escalas e oclusão deixa as suas possibilidades de aplicação um tanto mais restritas. Por fim, observando os melhores resultados obtidos, observa-se que o conjunto *Wider Face* é bem desafiador e mesmo técnicas mais recentes como a YOLOv5 ainda possuem grande margem para ganho, sendo necessário, nestes casos, utilizar métodos melhores que, entretanto, podem necessitar de um maior poder computacional.

4.1.2 Comparação de tempo de processamento

Juntamente com o experimento anterior, também foi analisada a eficiência de cada um dos métodos, verificando o tempo médio de inferência por imagem de cada um durante a execução das detecções. As predições foram realizadas em dois ambientes diferentes descritos na Seção 3.2.2. De modo geral, a diferença entre os ambientes está na disponibilidade de núcleos CUDA.

Os resultados obtidos são apresentados na Tabela 6. Em ambas as máquinas, o método Viola-Jones apresentou desempenho similar, com tempo de detecção por imagem imagem

Figura 32 – Detecção Viola-Jones somente frontal

Fonte: Produção própria do Autor.

Tabela 6 – Tempo médio de predição de cada método

Método	**com CUDA**	**sem CUDA**
Yolo	10	112
Viola-Jones	30	31

Fonte: Produção do próprio autor

próximo a 30 ms. Este resultado foi semelhante, pois a implementação do algoritmo não utiliza os núcleos CUDA. Por ser independente da disponibilidade de núcleos CUDA torna sua utilização para sistemas embarcados bem interessante.

Por outro lado, por utilizar PyTorch e ser otimizado para utilização de núcleos CUDA, o método YOLO teve um desempenho cerca de 11 vezes melhor no sistema que os possui e seu resultado de 10 ms por imagem o torna muito interessante para sistemas mais robustos que sejam em tempo real.

Assim, considerando um sistema de tempo real, não existe uma definição exata de quantos quadros por segundo, também conhecido por frames por segundo (FPS), são necessários. Existem casos diversos, podendo variar de 30 quadros até mesmo 1 quadro por segundo dependendo da sua aplicação, sendo inclusive comum escolher uma taxa menor de quadros para poder aumentar a definição das imagens processadas e assim obter resultados mais precisos. Para os métodos apresentados, tanto o Viola-Jones quanto a YOLOv5 apresentam taxas de quadros por segundo bem elevadas, sendo elas 33 e 100, respectivamente, abrindo possibilidades para diversas aplicações.

5 CONCLUSÃO E TRABALHOS FUTUROS

O objetivo principal deste trabalho foi compreender a diferença entre técnicas clássicas e mais recentes de detecção facial. Para isso, optou-se pela utilização de dois métodos conhecidos, o Viola-Jones e o YOLO para realizar comparações de arquitetura e capacidade e dois conjuntos consolidados de imagens: a *Wider Face* e a *Face Detection Data Set and Benchmark*.

Para o método Viola-Jones foi escolhido o modelo já pré-treinado disponibilizado na biblioteca OpenCV e para a YOLO foi necessário treinar um modelo YOLOv5 para detecção de faces utilizando o conjunto de dados de treino da *Wider Face*. Durante os testes foram realizadas predições para a verificação dos desempenhos dos métodos. Em adição, fez-se uma análise da velocidade de cada algoritmo para dois ambientes diferentes de execução, com e sem a presença de núcleos CUDA.

As técnicas selecionadas foram avaliadas utilizando 4 métricas diferentes. Para o conjunto de dados de validação da *Wider Face* foi obtida uma acurácia de 16,56% (Viola-Jones) e 50,41% (YOLO) e no conjunto de dados FDDB uma acurácia de 64,27% (Viola-Jones) e 85,36% (YOLO). Os valores demonstram a superioridade dos algoritmos baseados em redes neurais sobre métodos mais antigos, principalmente com relação a capacidade de adaptação para detecção de faces em diferentes situações.

Todavia, é importante destacar que o resultado do Viola-Jones no conjunto FDDB (Tabela 4) demonstra sua aplicabilidade principalmente se somado à sua simplicidade e não ser otimizado para algum tipo de hardware demonstrado na Tabela 6. Isso é evidenciado pela diferença dos resultados nos conjuntos de dados, para faces mais facilmente detectáveis, ou seja, em posição frontal, sem oclusão e com tamanho razoável o método apresenta desempenhos melhores como apresentado por Basbrain, Gan e Clark (2017).

É interessante ressaltar também que já existem estudos e projetos que visam a utilização de métodos baseados em redes neurais em dispositivos que não possuem unidade gráfica de processamento (JAHANSHAHI, 2019), o que poderia melhorar os resultados da rede YOLO em sua velocidade de processamento para os casos onde não existe disponibilidade de núcleos CUDA.

Para trabalhos futuros é interessante mencionar que uma comparação mais precisa pode ser obtida se o conjunto de imagens for separada de acordo com os níveis de dificuldade

de detecção dos rostos em cada imagem, separando em conjuntos como: Com oclusão, iluminação alterada, posição não frontal e expressão alterada. Outra abordagem possível também seria adicionar diferentes métodos para realizar uma comparação mais abrangente e destacar as diferentes vantagens de cada método. Métodos mais avançados para detecção de faces, como TinaFace, AInnoFace e a Face R-FCN, também podem ser avaliados.

REFERÊNCIAS

AL-ALLAF, O. N. A. Review of face detection systems based artificial neural networks algorithms. The International Journal of Multimedia Its Applications (IJMA), v. 6, n. 1, 2014. Citado na página 16.

AYDIN, I.; OTHMAN, N. A. A new iot combined face detection of people by using computer vision for security application. In: 2017 International Artificial Intelligence and Data Processing Symposium (IDAP). [S.l.: s.n.], 2017. p. 1–6. Citado na página 12.

BARTLETT, M. S.; LITTLEWORT, G.; FASEL, I.; MOVELLAN, J. R. Real time face detection and facial expression recognition: Development and applications to human computer interaction. In: 2003 Conference on Computer Vision and Pattern Recognition Workshop. [S.l.: s.n.], 2003. v. 5, p. 53–53. Citado na página 12.

BASBRAIN, A.; GAN, J.; CLARK, A. Accuracy enhancement of the viola-jones algorithm for thermal face detection. In: Intelligent Computing Methodologies. [S.l.: s.n.], 2017. p. 71–82. ISBN 978-3-319-63314-5. Citado na página 45.

BOCHKOVSKIY, A.; WANG, C.; LIAO, H. M. Yolov4: Optimal speed and accuracy of object detection. CoRR, abs/2004.10934, 2020. Disponível em: <https://arxiv.org/abs/2004.10934>. Citado na página 16.

EDWARDS, C. Deep Learning Hunts for Signals Among the Noise. 2018. 13-14 p. Disponível em: <https://cacm.acm.org/magazines/2018/6/228030-deep-learning-hunts-for-signals-among-the-noise/fulltext>. Acesso em: 17 nov. 2021. Citado na página 22.

EVERINGHAM, M.; GOOL, L. V.; WILLIAMS, C.; WINN, J.; ZISSERMAN, A. The pascal visual object classes (voc) challenge. International Journal of Computer Vision, v. 88, p. 303–338, 06 2010. Citado na página 34.

GOEDERT, M. L.; FILHO, P. L. P.; BLANCO, D. R. Natural computing: concepts and applications of computing inspired by nature. ESPACIOS, v. 38, n. 34, 2017. Citado na página 21.

HAYKIN, S. S. Neural networks and learning machines. Third. Upper Saddle River, NJ: Pearson Education, 2009. Citado na página 21.

HUANG, G.; LIU, Z.; MAATEN, L. van der; WEINBERGER, K. Q. Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2017. Citado 2 vezes nas páginas 27 e 28.

ISMAIL, L.; SHAMSUDDIN, S.; YUSSOF, H.; HASHIM, H.; BAHARI, S.; JAAFAR, A.; ZAHARI, I. Face detection technique of humanoid robot nao for application in robotic assistive therapy. In: 2011 IEEE International Conference on Control System, Computing and Engineering. [S.l.: s.n.], 2011. p. 517–521. Citado na página 12.

JAHANSHAHI, A. Tinycnn: A tiny modular CNN accelerator for embedded FPGA. CoRR, abs/1911.06777, 2019. Disponível em: <http://arxiv.org/abs/1911.06777>. Citado na página 45.

JAIN, V.; LEARNED-MILLER, E. FDDB: A Benchmark for Face Detection in Unconstrained Settings. Amherst, MA 01003, United States, 2010. 11 p. Citado 3 vezes nas páginas 13, 32 e 33.

KAMALI, K. Deep Learning (Part 3) - Convolutional neural networks (CNN). 2016. Disponível em: <https://training.galaxyproject.org/training-material/topics/statistics/tutorials/CNN/tutorial.html#DumoulinVisin>. Acesso em: 04 dez. 2021. Citado na página 23.

KARN, U. An Intuitive Explanation of Convolutional Neural Networks. 2016. Disponível em: <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>. Acesso em: 01 dez. 2021. Citado 3 vezes nas páginas 22, 23 e 24.

KUMAR, A.; KAUR, A.; KUMAR, M. Face detection techniques: a review. Artificial Intelligence Review, v. 52, p. 927–948, 2018. Citado na página 16.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. nature, Nature Publishing Group, v. 521, n. 7553, p. 436, 2015. Citado na página 22.

LEE, S. Understanding Face Detection with the Viola-Jones Object Detection Framework. 2020. Disponível em: <https://towardsdatascience.com/understanding-face-detection-with-the-viola-jones-object-detection-framework-c55cc2a9da14>. Acesso em: 22 nov. 2021. Citado na página 17.

LIN, T.-Y.; DOLLáR, P.; GIRSHICK, R. B.; HE, K.; HARIHARAN, B.; BELONGIE, S. J. Feature pyramid networks for object detection. In: CVPR. IEEE Computer Society, 2017. p. 936–944. ISBN 978-1-5386-0457-1. Disponível em: <http://dblp.uni-trier.de/db/conf/cvpr/cvpr2017.html#LinDGHHB17>. Citado na página 29.

LIU, S.; QI, L.; QIN, H.; SHI, J.; JIA, J. Path aggregation network for instance segmentation. CoRR, abs/1803.01534, 2018. Disponível em: <http://arxiv.org/abs/1803.01534>. Citado na página 30.

MUJTABA, H. Face Detection using Viola Jones Algorithm. 2020. Disponível em: <https://www.mygreatlearning.com/blog/viola-jones-algorithm/>. Acesso em: 15 nov. 2021. Citado 2 vezes nas páginas 17 e 18.

NELSON, J. Responding to the Controversy about YOLOv5. 2020. Disponível em: <https://blog.roboflow.com/yolov4-versus-yolov5/>. Acesso em: 20 jan. 2022. Citado na página 25.

NIELSEN, M. A. misc. Neural Networks and Deep Learning. Determination Press, 2018. Disponível em: <http://neuralnetworksanddeeplearning.com/>. Citado na página 24.

O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. CoRR, abs/1511.08458, 2015. Disponível em: <http://dblp.uni-trier.de/db/journals/corr/corr1511.html#OSheaN15>. Citado na página 22.

PASZKE, A.; GROSS, S.; MASSA, F.; LERER, A.; BRADBURY, J.; CHANAN, G.; KILLEEN, T.; LIN, Z.; GIMELSHEIN, N.; ANTIGA, L.; DESMAISON, A.; KÖPF, A.; YANG, E. Z.; DEVITO, Z.; RAISON, M.; TEJANI, A.; CHILAMKURTHY, S.; STEINER, B.; FANG, L.; BAI, J.; CHINTALA, S. Pytorch: An imperative style, high-performance deep learning library. CoRR, abs/1912.01703, 2019. Disponível em: <http://arxiv.org/abs/1912.01703>. Citado na página 34.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You only look once: Unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2016. p. 779–788. Citado 4 vezes nas páginas 12, 25, 26 e 27.

RIZVI, D. Q. A review on face detection methods. Journal of Management Development and Information Technology, v. 11, 02 2011. Citado na página 13.

SZELISKI, R. Computer vision algorithms and applications. London; New York: Springer, 2011. ISBN 9781848829343 1848829345 9781848829350 1848829353. Disponível em: <http://dx.doi.org/10.1007/978-1-84882-935-0>. Citado 2 vezes nas páginas 17 e 19.

THUAN, D. Evolution of Yolo algorithm and Yolov5: The State-of-the-Art object detention algorithm. Yliopistokatu 9, 90570 Oulu, Finland, 2021. Citado 2 vezes nas páginas 15 e 26.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. [S.l.: s.n.], 2001. v. 1, p. I–I. Citado 4 vezes nas páginas 12, 17, 18 e 20.

WANG, C.; LIAO, H. M.; YEH, I.; WU, Y.; CHEN, P.; HSIEH, J. Cspnet: A new backbone that can enhance learning capability of CNN. CoRR, abs/1911.11929, 2019. Disponível em: <http://arxiv.org/abs/1911.11929>. Citado 2 vezes nas páginas 27 e 28.

WANG, Y.; JI, X.; ZHOU, Z.; WANG, H.; LI, Z. Detecting faces using region-based fully convolutional networks. CoRR, abs/1709.05256, 2017. Disponível em: <http://arxiv.org/abs/1709.05256>. Citado na página 12.

YANG, S.; LUO, P.; LOY, C. C.; TANG, X. Wider face: A face detection benchmark. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2016. Citado 3 vezes nas páginas 13, 31 e 32.

ZHANG, F.; FAN, X.; AI, G.; SONG, J.; QIN, Y.; WU, J. Accurate face detection for high performance. CoRR, abs/1905.01585, 2019. Disponível em: <http://arxiv.org/abs/1905.01585>. Citado na página 12.

ZHU, Y.; CAI, H.; ZHANG, S.; WANG, C.; XIONG, Y. Tinaface: Strong but simple baseline for face detection. CoRR, abs/2011.13183, 2020. Disponível em: <https://arxiv.org/abs/2011.13183>. Citado na página 12.

ZIMMERMANN, K. A. History of Computers: A Brief Timeline. 2017. Disponível em: <https://www.livescience.com/20718-computer-history.html>. Acesso em: 15 nov. 2021. Citado na página 12.

Anexos

.1 Códigos e Bibliotecas Utilizadas

Durante o projeto, foram utilizados códigos de diferentes bibliotecas e repositórios. Seguem as referências.

- **Repositório da YOLOv5:** https://github.com/ultralytics/yolov5
- **Conversão das anotações do FDDB:** https://github.com/hualitlc/MTCNN-on-FDDB-Dataset/blob/master/convertEllipseToRectangle.py
- **Cálculo das métricas dos experimentos:** https://github.com/rafaelpadilla/review_object_detection_metrics
- **OpenCV para Python:** https://pypi.org/project/opencv-python/
- **Repositorio do projeto:** https://github.com/lucasff57/Viola_x_Yolo