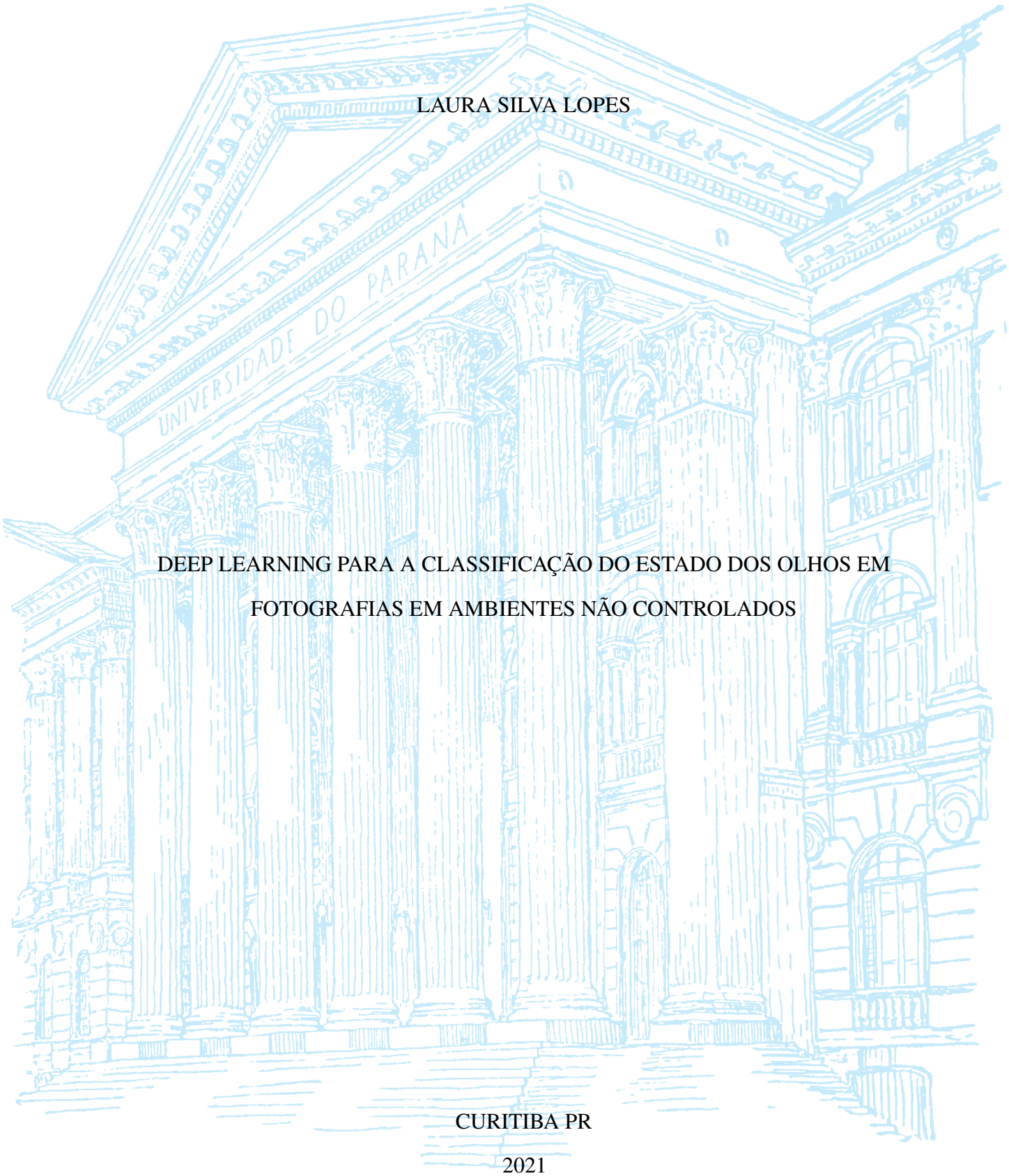
UNIVERSIDADE FEDERAL DO PARANÁ

LAURA SILVA LOPES



DEEP LEARNING PARA A CLASSIFICAÇÃO DO ESTADO DOS OLHOS EM FOTOGRAFIAS EM AMBIENTES NÃO CONTROLADOS

CURITIBA PR

2021

LAURA SILVA LOPES

DEEP LEARNING PARA A CLASSIFICAÇÃO DO ESTADO DOS OLHOS EM FOTOGRAFIAS EM AMBIENTES NÃO CONTROLADOS

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: David Menotti.

CURITIBA PR

2021

À minha família.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus, que sempre foi e é meu refúgio, paz e fonte da minha determinação para superar os obstáculos. Aos meus pais, Donizete e Angela, pelo apoio e amor incondicional que sempre me deram. Vocês são meu exemplo de vida e companheirismo. Aos meus padrinhos e prima, Anilce, Gilberto e Isabella, por me acolherem nos primeiros anos de faculdade e serem minha família em Curitiba. A minha irmã e cunhado, Bruna e Lucas, pelo exemplo de profissionais e por não medirem esforços em me ajudar a realizar esse e muitos outros sonhos. Dedico a vocês também essa conquista. A minha irmã, Carla, pelo companheirismo, inspiração, por ser minha confidente e força nos momentos difíceis. Aos mestres e amigos Professores, em especial ao meu orientador David Menotti, pelos ensinamentos e conselhos dados. Vocês são fonte de inspiração e foram essenciais para a minha formação como pessoa e profissional. Aos colegas e amigos de curso pelas alegrias, tristezas, sucessos e fracassos compartilhados nesses anos que estivemos juntos.

Meu muito obrigada a todos.

RESUMO

Identificar sonolência em motoristas, controlar dispositivos eletrônicos com os olhos e detectar pessoas com olhos fechados em fotografias são algumas das aplicações mais frequentes da classificação do estado dos olhos. Essa classificação geralmente é feita em duas etapas diferentes, uma encarregada de detectar a região dos olhos e outra para classificar o seu estado em aberto ou fechado. Entretanto, há a possibilidade de diminuir o tempo de processamento, quantidade de operações realizadas e a complexidade computacional do modelo com apenas uma etapa. Portanto, o objetivo desse trabalho é estudar técnicas recentes de *Deep Learning* para detecção de objetos, em especial YOLOv4, e usá-la para a detecção e classificação do estado dos olhos simultaneamente em uma única rede neuronal. Comparou-se esse método com outro que realiza a detecção dos olhos e classificação do seu estado em etapas e redes neurais diferentes, YOLOv4 e ResNet-50. Foram anotadas novas imagens capturadas em ambientes não controlados, com grande variedade no número de indivíduos, suas posições na imagem, contraste de cor, iluminação, entre outros, para serem utilizadas nos experimentos. Os resultados indicam que o experimento realizado em duas etapas obteve melhor desempenho que aquele que realiza a detecção e classificação simultaneamente com a YOLOv4. Além disso, a partir de uma avaliação visual foi possível perceber situações onde os modelos falham em classificar corretamente o estado dos olhos, como em casos de oclusão parcial da face e/ou olhos, pose não habitual do indivíduo e uso de maquiagem marcante. Este estudo apresenta a fundamentação teórica de *Deep Learning*, detectores de objetos e métricas de avaliação, uma breve revisão de trabalhos relacionados, metodologias dos experimentos realizados e resultados.

Palavras-chave: Estado dos olhos, Classificação, *Deep Learning*, CNN, YOLOv4

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

AP	Precisão média
BBox	*bounding box*
BBoxes	*bounding boxes*
CNN	Redes Neuronais Convolucionais
CSP	*Cross-Stage-Partials*
CVAT	*Computer Vision Annotation Tool*
EAR	*Eye Aspect Ratio*
FC	*Fully-Connected*
FN	Falso Negativo
FP	Falso Positivo
FPN	Rede de Características Piramidal
FPS	*Frames* por segundo
GPU	*Graphics Processing Unit*
GT	*Ground-Truth*
IoU	Interseção sobre União
mAP	Média das precisões médias
MTCNN	*Multitask Convolutional Neural Networks*
PANet	*Path Aggregation Network*
PP-YOLO	*PaddlePaddle-YOLO*
ReLU	*Rectified Linear Function*
ResNet	*Residual Networks*
RF	*Receptive Fields*
ROI	Regiões de Interesse
SSD	*Single Shot MultiBox Detector*
SVM	*Support Vector Machine*
TN	Verdadeiro Negativo
TP	Verdadeiro Positivo
YOLO	*You Only Look Once*

SUMÁRIO

1 INTRODUÇÃO

A classificação do estado dos olhos é uma área ainda em estudo e de diversas importâncias. Com essa classificação, é possível identificar o piscar de olhos em vídeos, permitindo sinalizar sonolência ou distração de motoristas na tentativa de evitar acidentes de trânsito, ou permitir que pessoas com restrições motoras possam navegar em interfaces gráficas e controlar dispositivos eletrônicos através dos olhos. No campo da imagem estática, essa classificação também é útil, por exemplo, para detectar pessoas que estejam com os olhos fechados e sinalizar para que uma nova foto seja tirada.

Estudos mais antigos utilizam técnicas simples de processamento de imagens para essa classificação, como detectar olhos por correspondência a templates e máscaras e o seu estado por uma análise em nível de pixel da imagem. Porém, não são técnicas robustas à ambientes e condições não controladas como a pose do indivíduo ou a qualidade da imagem. Além de não serem técnicas tão eficientes em aplicações de tempo real quanto as de *deep learning*, estudadas neste trabalho.

A grande maioria dos trabalhos dividem a classificação em pelo menos duas etapas, primeiro detecta-se a região dos olhos e então os classifica em fechados ou abertos. Mesmo que classificar dessa forma seja eficiente, há também a discussão de se realizar as duas etapas simultaneamente e num mesmo modelo de rede. Assim, o tempo necessário de processamento e a quantidade de operações realizadas são reduzidos, diminuindo também a complexidade computacional do detector. O desafio de realizar a classificação em apenas uma etapa é o de manter ou, idealmente, aumentar a acurácia dos modelos de duas etapas comumente utilizados.

Para que o detector possa aprender de forma genérica e seja robusto as mais diversas variações possíveis de um mesmo objeto, um *dataset* completo de imagens é necessário. Essa completude é dada pela quantidade de imagens e o equilibrio e variedade das representações de cada classe que se deseja detectar. *Datasets* que possuem muitas imagens mas pouca variedade fazem com que o detector tenha um excesso de confiança para determinadas características, memorizando e não aprendendo de fato a classificar as representações. Esse fenômeno, chamado de sobreajuste, ocorre por que o detector se adapta aos dados com os quais está sendo treinado e, portanto, não os generaliza. Em contrapartida, quando o *dataset* possui boa variedade porém não há representações suficientes, configura o chamado subajuste. Isso também acarreta em desempenho ruim nos testes por que o modelo não consegue se adaptar sequer aos dados com os quais foi treinado.

Algumas técnicas são usadas para melhorar a representação do *dataset* e evitar que o modelo se torne enviesado ou insuficiente para a detecção. Uma das mais utilizadas é o *data augmentation* que gera novas imagens a partir daquelas já disponíveis, aumentando o tamanho e a diversidade do *dataset*. Por usar imagens já anotadas, essa técnica reduz o tempo

que seria gasto em anotar novas imagens. São criadas cópias com pequenas modificações de iluminação, contraste, enfoque, escala, rotação, inclinação e/ou posicionamento da representação na imagem. Também é possível ser feita a combinação de imagens, ou seja, recortes de uma imagem concatenadas a outra, que fazem com que o detector dê mais importância a outras características do objeto. Quanto mais diversificado for o *dataset* final, mais preciso e genérico poderá ser o detector.

Além da necessidade de garantir a completude do *dataset*, é preciso encontrar, ou construir, o ideal para determinada aplicação. Para a classificação do estado dos olhos, estão a disposição *datasets* com características diversas de vídeos ou imagens, ilustrados na Figura 1.1. CEW, 300-W, BioID e GI4E são exemplos de *datasets* de fotos com apenas o rosto de um indivíduo em ambientes não controlados, ou seja, com variedade de iluminação, pose e fundo. Nesses exemplos são anotados pontos de interesse na face, posição dos olhos e seu estado. RT-BENE e MRL são exemplos de imagens dos olhos apenas, sendo o segundo obtido com diferentes câmeras infravermelho. Entretanto, esses *datasets* não possuem grande variedade, principalmente, do número de indivíduos envolvidos, uma das características importantes para o que se pretende aplicar este trabalho.

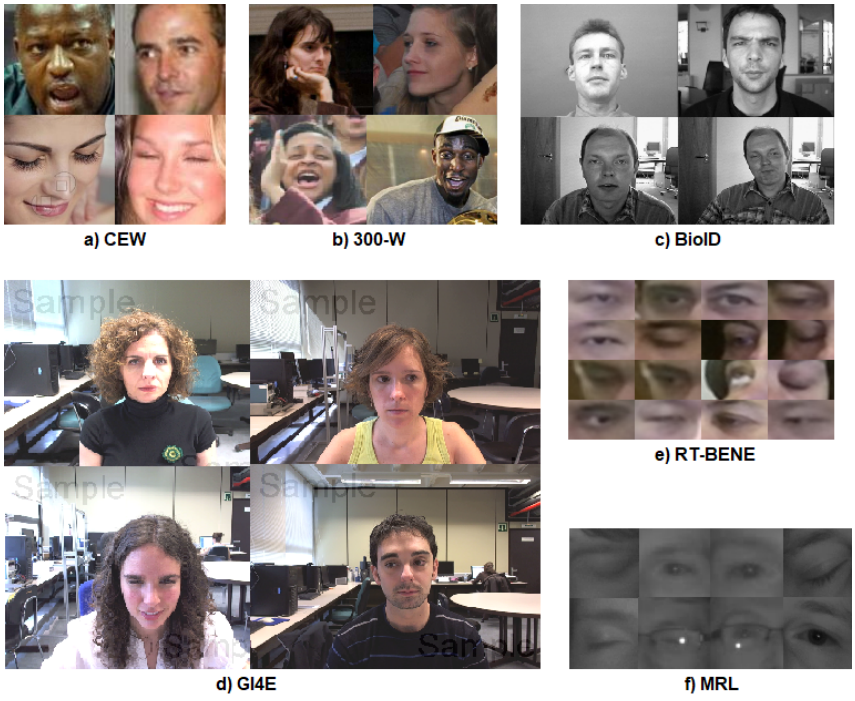


Figura 1.1: Exemplos de imagens de *datasets* disponíveis para a classificação do estado dos olhos.

Entendendo os desafios expostos, este trabalho tem por objetivo principal **detectar e classificar simultaneamente o estado dos olhos em fotografias capturadas em ambientes não controlados através de técnicas de *deep learning***. Para que isso seja atingido, é proposta a anotação de olhos fechados e abertos em uma seleção de imagens do *dataset* WIDERFACE, composto por fotografias em ambientes não controlados retiradas da *internet*. Essa seleção foi feita a partir das faces já anotadas pela equipe que propôs o *dataset* original. Foram selecionadas

imagens que contém faces com mais de 40 pixels de altura, o mínimo determinado para que o estado dos olhos seja detectável por humanos. Com o *dataset* de imagens proposto, serão avaliados e comparados os desempenhos das classificações do estado dos olhos em uma e duas etapas diferentes utilizando YOLOv4 e ResNet-50.

O restante do trabalho está organizado da seguinte forma: no Capítulo 2, encontra-se a fundamentação teórica de *deep learning*, redes neuronais convolucionais, detectores de objetos (e.g. YOLOv4) e métricas de avaliação; os trabalhos relacionados são brevemente apresentados no Capítulo 3; os métodos e recursos utilizados para este trabalho são detalhados no Capitulo 4; os resultados dos experimentos são apresentamos e discutidos no Capítulo 5; e, por fim, o Capítulo 6 conclui este estudo.

2 FUNDAMENTAÇÃO TEÓRICA

Deep Learning é uma vertente do aprendizado de máquina, inspirado no funcionamento do cérebro humano, que permite que o computador aprenda representações [1] por meio da experiência e treinamento. É uma técnica usada principalmente nas áreas de visão computacional, processamento de imagens e computação gráfica. Os modelos de *deep learning* são compostos de vários estágios de processamento que recebem uma entrada de dados e a transforma em uma representação mais abstrata (LeCun et al., 2015) para que aspectos importantes na classificação de um objeto sejam amplificados e aspectos irrelevantes sejam suprimidos (LeCun et al., 2015). Há inumeras possibilidades para aplicações de técnicas de *deep learning*, porém, este trabalho foca em introduzir os conceitos aplicados na classificação de objetos em imagens.

2.1 REDES NEURONAIS CONVOLUCIONAIS

Um dos modelos mais utilizados para a detecção e/ou classificação de objetos é a rede neuronal convolucional (CNN, do inglês *Convolutional Neural Network*), desenhada para processar dados no formato de matrizes de 2 ou 3 dimensões (LeCun et al., 2015), como as imagens. Essas dimensões correspondem à altura, largura e "profundidade", ou seja, canais do sistema de cores da imagem RGB ou outro (Dettmers, 2015). Esse modelo é estruturado basicamente em dois grandes estágios, o aprendizado de características e a classificação de objetos. O primeiro é composto por, principalmente, dois tipos de camadas de processamento (ou séries de operações), convolucionais e *pooling*, seguidas de camadas *Fully-Connected* (FC) para o estágio seguinte, classificação. A Figura 2.1 ilustra uma arquitetura de CNN com 2 camadas de convolução, 2 de *pooling* e 2 FC usada para classificar a imagem de um leão.

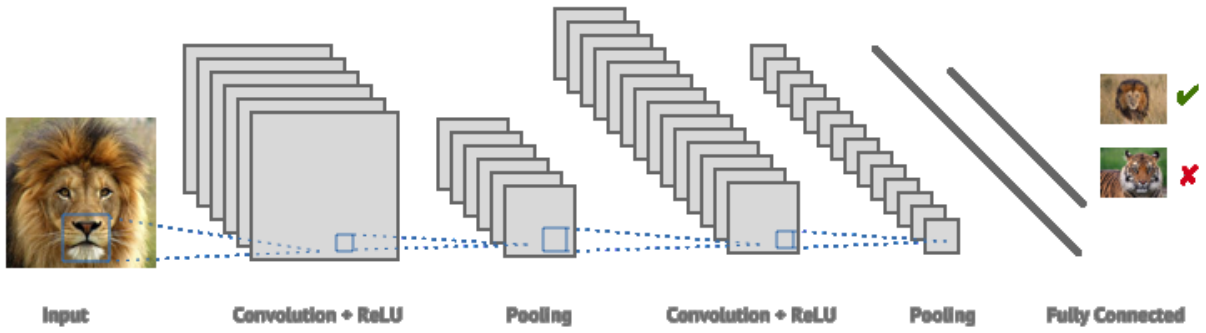


Figura 2.1: Exemplo de arquitetura CNN composta de 2 camadas de convolução e 2 de *pooling* para o estágio de aprendizado de características e 2 camadas *fully connected* para o estágio de classificação. Fonte Tejani (2016)

[1] REPRESENTAÇÃO: Imagem ou ideia que traduz nossa concepção de alguma coisa ou do mundo. In: Michaelis (2015)

Nas camadas convolucionais, ilustrada na Figura 2.2, a matriz de pixels de entrada é dividida em pedaços menores, chamados de *Receptive Fields* (RF) e possuem filtros de mesma dimensão destes pedaços. Estes filtros são matrizes de valores reais (chamados de parâmetros ou pesos) iniciados de forma aleatória mas ajustáveis durante toda a fase de treinamento. Os filtros são usados para detectar características específicas numa imagem, sejam elas bordas simples ou estruturas complexas como olhos e narizes (Tejani, 2016). É calculada a combinação linear do RF com a matriz de filtro e aplicada uma função de ativação não linear ao resultado, e.g. ReLU, Mish. Os filtros percorrem toda a imagem de entrada e são combinados a cada RF a um passo, ou *stride*, pré-determinado e, o valor resultante de cada um corresponde a um pixel no chamado "mapa de características"usado como entrada para a camada seguinte (Dettmers, 2015).

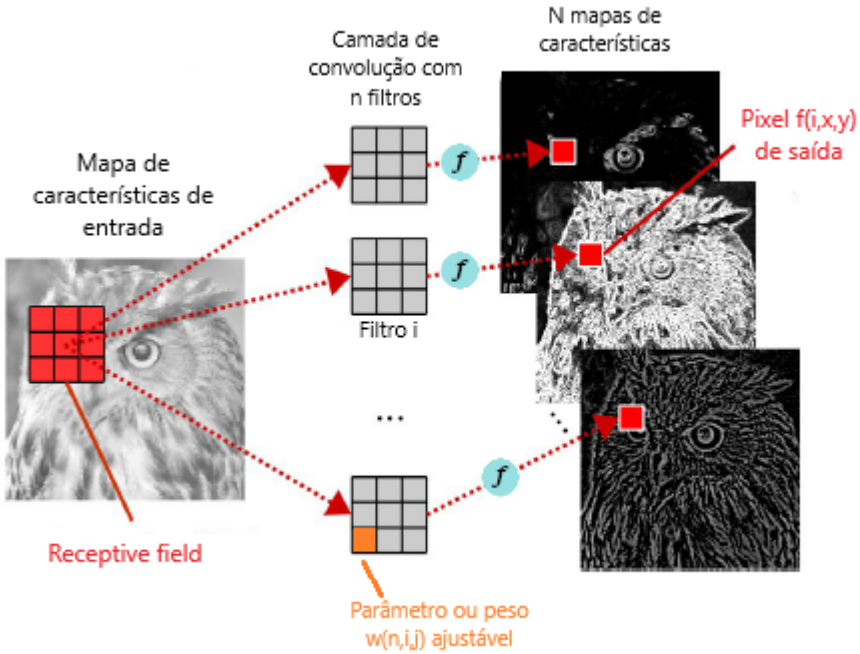


Figura 2.2: Na camada de convolução são aplicados filtros (matrizes de parâmetros ajustáveis) em porções da imagem de entrada, chamadas de RF. A saída do filtro em cada uma dessas porções corresponde a um pixel no mapa de características usado como entrada para a próxima camada. Os círculos em azul correspondem à função de ativação (e.g. ReLU). Adaptado de Ponti et al. (2017).

A camada de *pooling*, processada antes ou depois das camadas convolucionais, serve para reduzir a dimensão da entrada para a próxima camada e, consequentemente, reduzir a complexidade computacional da rede. Isso é feito por meio de operações em regiões dos mapas de características que corresponderão a um elemento no mapa reduzido cada. Exemplos de operações utilizadas nesse processo e ilustradas na Figura 2.3 são: *Average Pooling*, que calcula o valor médio dos pixels dentro desta região; e *Max Pooling*, onde o maior valor da região é mantido (Tejani, 2016). Nas camadas de convoluções e *pooling*, os dados de saída continuam sendo matrizes, porém, em dimensões menores das camadas anteriores e com um nível de abstração maior.

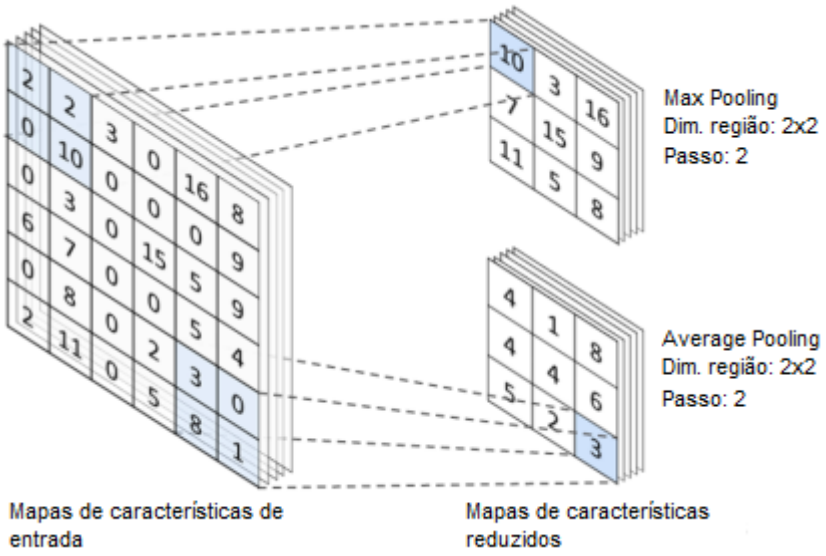


Figura 2.3: Nas camadas de *pooling* os mapas de característas são reduzidos por *Max Pooling* ou *Average Pooling*. Adaptado de Voinov (2020).

Ilustradas na Figura 2.4, as camadas *Fully-Connected* (FC), diferente das camadas explicadas anteriormente, produzem apenas um valor ou um vetor de valores escalares que representam a propabilidade da representação detectada pertencer a uma classe de objetos. As arquiteturas de redes neuronais podem conter uma ou mais camadas FC, sendo a última a responsável por gerar a classificação final. Os mapas de características são vetorizados para que cada elemento desse vetor seja conectado a cada elemento da primeira FC. Entre as camadas também são utilizadas funções de ativação e, por fim, a operação para classificar as características extraídas nas camadas de convolução anteriormente, e.g. *softmax* (Voinov, 2020). A classe de maior probabilidade é então atribuída ao objeto detectado e a rede neuronal completa o processamento da imagem.

Há dois tipos de aprendizados na fase de treinamento da rede: o supervisionado, onde cada entrada usada já possui uma saída esperada; e o não supervisionado, onde são feitas algumas suposições sobre o dado para a classificação (Goodfellow et al., 2016). O treinamento no aprendizado supervisionado consiste de ciclos, ou *batches*, em que a rede é alimentada com novas representações do *dataset*. No final de cada ciclo, os parâmetros dos filtros são ajustados através de algoritmos como o *backpropagation* a fim de minimizar a diferença entre a saída obtida e a esperada (Voinov, 2020). É formada uma época quando todas as imagens de treinamento foram usadas como entrada para a rede neuronal.

Na fase de testes, após a de treinamento, a rede é alimentada com imagens ainda não utilizadas para certificar que a rede pôde aprender sozinha as representações e suas características e, assim, classificar corretamente o objeto. Um bom detector de objetos deve ser rápido, preciso, capaz de reconhecer uma vasta variedade de objetos e ser invariante às condições da imagem, e, com a introdução das redes neuronais convolucionais, isso tem sido cada vez mais possível.

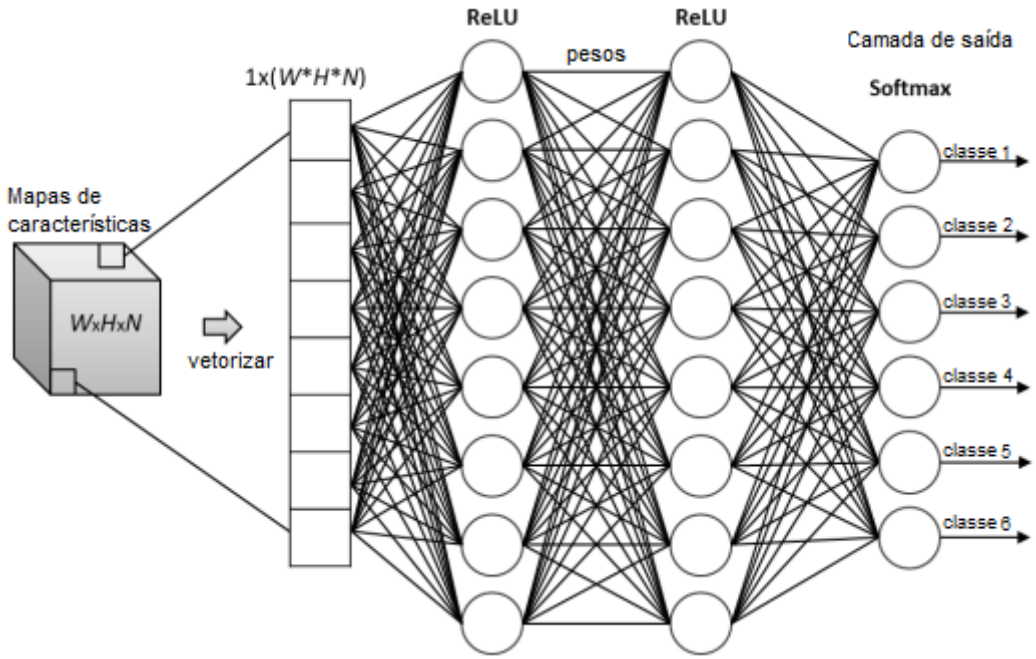


Figura 2.4: Nas camadas *Fully-Connected*, os mapas de características são vetorizados de modo que cada elemento do vetor esteja conectado a cada elemento das camadas. Esses elementos passam por funções de ativação e, por fim, uma operação para a classificação final. Adaptado de Voinov (2020).

2.1.1 ResNet

Embora pareça que aumentar o número de camadas na rede neuronal melhora o aprendizado, não é o que realmente acontece. Quanto mais camadas o modelo possui, dizemos que mais profundo ele é, e os gradientes de características da imagem se dissipam mais rapidamente entre as camadas. Essa distribuição dificulta o treinamento da rede, piora o seu desempenho e pode diminuir a acurácia a partir de determinado ponto por estar saturada (Boesch, 2021).

Para contornar esse problema, He et al. (2016) propõem uma estrutura de aprendizagem residual que facilita o treinamento, chamada ResNet. Essa estrutura possui conexões, também chamadas de atalhos, que pulam duas ou três camadas convolucionais ou FC, como ilustra a Figura 2.5. As conexões realizam a operação de soma de matrizes entre os mapas de características de mesma dimensão. Quando essa operação é feita entre camadas que possui mesma dimensão de saída, não são acrescentados parâmetros extras nem aumenta a complexidade computacional do modelo. Já entre camadas que não têm saídas de mesma dimensão, é preciso uma operação de convolução para igualá-las e então realizar a operação de soma.

Há ResNets com diferentes números de camadas, e.g. ResNet-18, ResNet-34, ResNet-50, ResNet-152, entre outras. Como o próprio nome sugere, os números são a quantidade de camadas neuronais que a ResNet possui. Portanto, a ResNet-50, um dos modelos utilizados neste trabalho, possui 50 camadas neuronais, sendo 48 convolucionais, 1 de *maxpool* e 1 *average pool*, com atalhos a cada 3 camadas. Esse modelo é um dos mais usados como *backbone* nos trabalhos de visão computacional ou outras áreas.

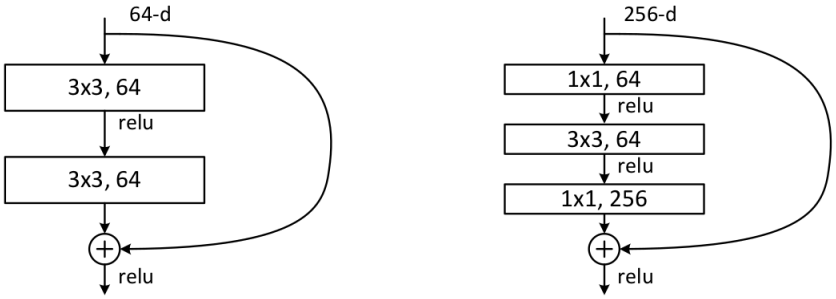


Figura 2.5: Ilustração dos atalhos entre as camadas da rede neuronal implementados pela ResNet. O atalho pode englobar 2 ou 3 camadas convolucionais. Fonte He et al. (2016).

2.2 DETECÇÃO DE OBJETOS

A detecção de objetos tem o objetivo de detectar a presença de diferentes objetos na imagem, encontrar suas localizações e classificá-los (Voinov, 2020). A rapidez do modelo e a sua acurácia em classificar corretamente o objeto devem estar equilibrados, e este é um dos desafios a se enfrentar nessa área de estudo. Além disso, os objetos podem estar em diferentes escalas, rotações e posições na imagem, o que dificulta ainda mais a detecção correta caso o detector não seja capaz de generalizar as representações.

Existem dois tipos de detectores, o de um e dois estágios. Naqueles de dois estágios, são selecionadas Regiões de Interesse (ROI) através de algoritmos como o *selective searching*, que agrupam pixels semelhantes da imagem. O detector desse tipo mais utilizado atualmente é a R-CNN e suas variantes. Os detectores de um estágio, que serão estudados de forma mais aprofundada nesse trabalho, não dependem de ROI para a detecção e, por isso, não possuem essa etapa. Esse tipo de detector é extremamente rápido e mais simples, porém, possui menor precisão comparado à detectores de dois estágios. Exemplos de detectores desse tipo são a YOLO e RetinaNet.

Depois de identificada as ROI, os detectores de dois estágios seguem da mesma forma que os de um estágio. O próximo passo, portanto, é a extração de características por uma CNN. Esse passo é também chamado de *backbone*, e é onde são extraídas das imagens informações necessárias para a detecção do objeto. *Backbones* muito utilizados na literatura são o VGG, GoogleNet, ResNet, DenseNet, entre outros.

Com o objetivo de contornar o problema de objetos poderem estar em diferentes escalas, o *backbone* possui inerente a extração de características em diversas dimensões da imagem. Redes com esse mecânismo geralmente incluem as pirâmides hierárquicas multi-escalas de mapas de características, chamada de FPN, do inglês, *Feature Pyramid Networks*. Essa rede compõe caminhos de cima para baixo, baixo para cima e conexões laterais entre as suas camadas. O caminho de baixo para cima é o comumente encontrado na CNN, onde, a medida que você avança "para cima"na rede, a dimensão da entrada diminui, aumentando o nível de abstração do mapa de características e o valor semântico da camada. O objetivo dessas combinações

de caminhos é de construir camadas de maior resolução a partir de camadas com maior valor semântico. A Figura 2.6 ilustra o fluxo de informações na FPN.

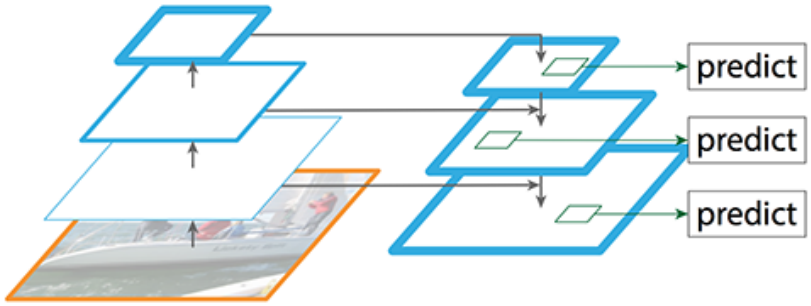


Figura 2.6: FPN combina o caminho de baixo para cima (convencional na CNN) e o caminho de cima para baixo com conexões laterais para contruir camadas de maior resolução a partir de camadas com maior valor semântico. Fonte Lin et al. (2016).

No próximo passo, chamado *neck*, todos os mapas de características de diferentes dimensões construídos a partir da FPN são combinados para servirem de entrada para o último passo do detector, o *head*, onde efetivamente são feitas as predições de classe e os objetos detectados são localizados. O resultado da classificação é usado para calcular a função de perda que será propagada do fim ao início da rede neuronal (processo chamado de *backpropagation*) ajustando todos os parâmetros de modo que o resultado obtido seja o mais próximo do esperado. Duas das funções mais utilizadas para calcular a acurácia do modelo são a *Mean Square Error* e *Mean Absolute Error* que calculam médias entre o valor obtido e o valor esperado.

2.2.1 You Only Look Once

Redmon et al. (2016) propuseram um sistema, chamado *You Only Look Once* (YOLO), que tem como objetivo transformar a detecção de objetos num problema de regressão (Redmon et al., 2016), ou seja, problema que prevê um valor númerico específico ou, neste caso, a classe a qual o objeto pertence. Como todo o processo de detecção é feito em uma só rede CNN, é um método extremamente rápido e, por isso, é usado em sistemas de imagens em tempo real. A YOLO analisa a imagem capturada inteiramente uma única vez em busca de objetos e não apenas regiões específicas como é feito em outros classificadores. Isso permite que o sistema aprenda representações genéricas de classes em diferentes contextos e, por permitir detectar um objeto por célula da matriz, garante diversidade nas previsões (Hui, 2018).

Na primeira versão do sistema, a imagem de entrada tem dimensão de 448 X 448 pixels dividida em uma matriz de S X S. Cada célula dessa matriz prevê até 2 *bounding boxes* (BBoxes) arbitrárias em torno dela e a probabilidade, ou valor de confiança, de conterem apenas um objeto de uma classe. Como um mesmo objeto pode estar contido em mais de uma *bounding box* (BBox), o algoritmo de *non-maximum suppression* é usado para eliminar detecções redundantes e reduzir os falsos positivos, deixando apenas aquela que mais se encaixa no objeto e que possui

um valor de confiança maior que 50% (Redmon et al., 2016). Todo esse processo é ilustrado na Figura 2.7.

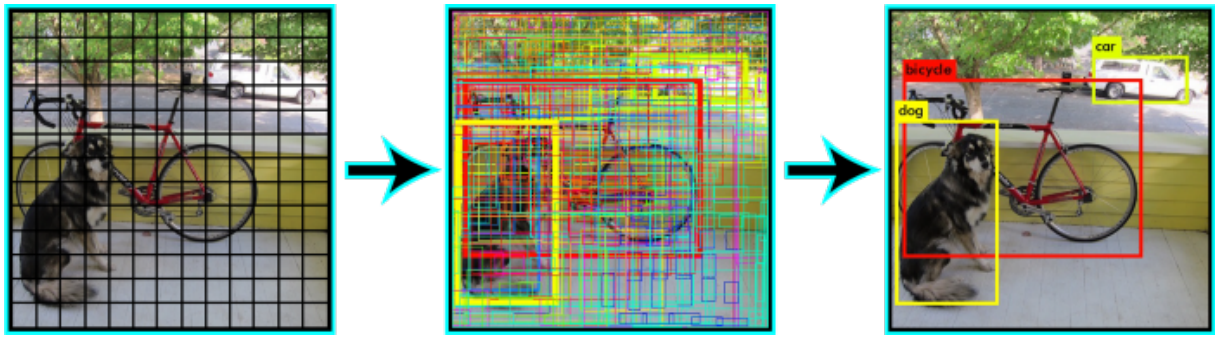


Figura 2.7: Visão geral do funcionamento da YOLO. A imagem de entrada é dividida em uma matriz de ordem S e cada célula prevê 2 *bounding boxes* arbitrárias, o valor de confiança de um objeto estar contido nestas caixas e a probabilidade deste pertencer a uma classe. *Non-maximum suppression* é então usado para descartar redundâncias e obter as detecções finais. Fonte Hui (2018)

Inspirada na arquitetura *GoogleNet* (Redmon et al., 2016), YOLO tem 24 camadas de convolução responsáveis por extrair características da imagem, seguidas de 2 camadas FC que calculam as probabilidades e coordenadas dos objetos detectados. A rede é ilustrada na Figura 2.8.

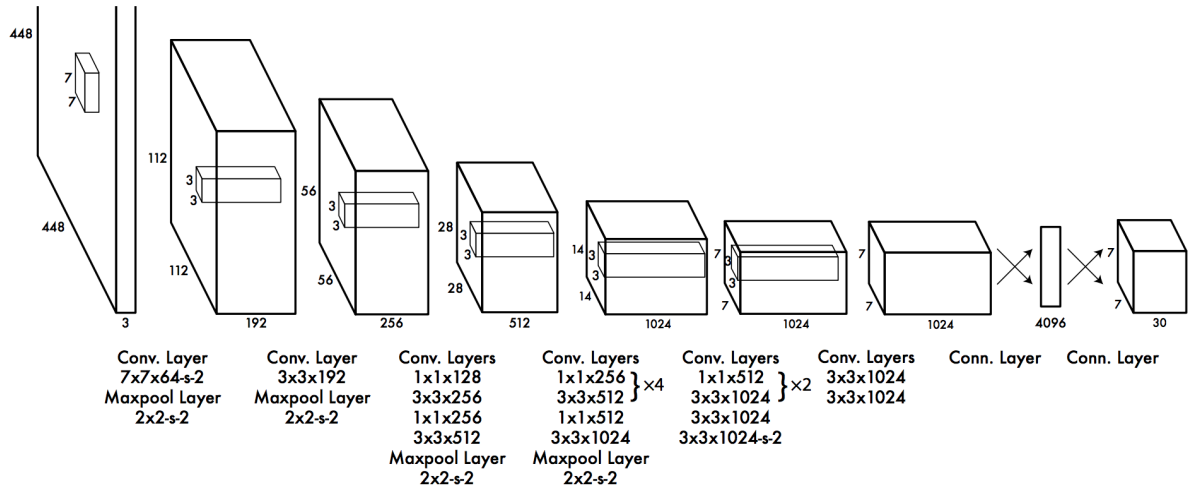


Figura 2.8: Arquitetura usada pela YOLO. Fonte Redmon et al. (2016)

YOLO atingiu uma acurácia de 64.3% no dataset Pascal VOC 2007, porém não é muito precisa na classificação de objetos pequenos, agrupados ou sobrepostos devido à limitação do número de caixas detectadas por célula de imagem (duas, podendo conter apenas um objeto) (Redmon et al., 2016).

YOLOv2, segunda versão do sistema apresentada por Redmon e Farhadi (2017), consegue detectar mais de 9000 classes de objetos em tempo real e tem como objetivo aprimorar a acurácia e corrigir os erros de localização de objetos na imagem da primeira versão. Na YOLOv2 são usadas formas já pré-definidas de BBoxes, chamadas de âncoras. Essas formas

foram definidas levando em conta que objetos de mesma classe possuem formatos semelhantes (Hui, 2018) (Figura 2.9) e, por ater em formatos específicos de caixas, deixa o processo de aprendizado mais simples para a rede. A posição e tamanho das caixas ajustadas nos objetos são previstos pelo *backbone* escolhido, Darknet-19.

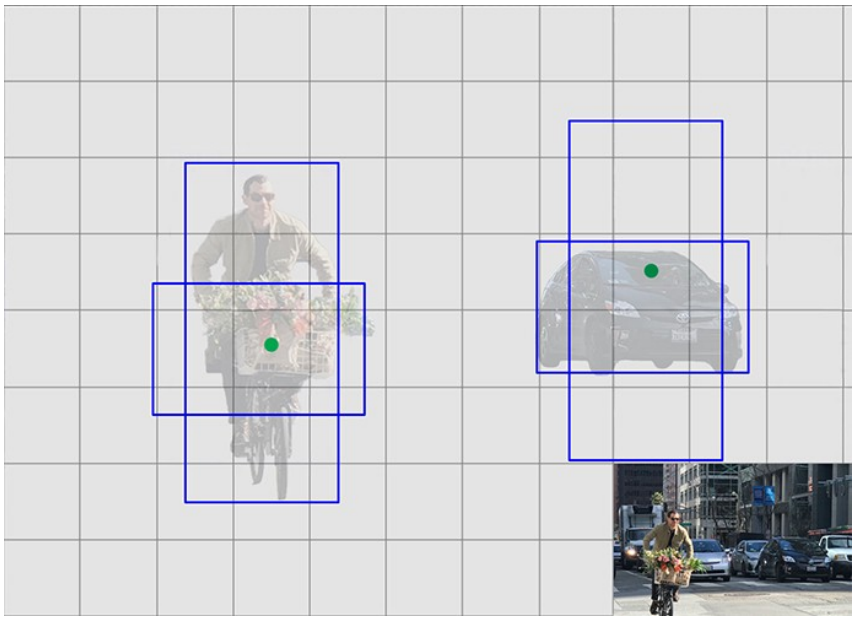


Figura 2.9: Ilustração de dois objetos e duas caixas âncoras que se encaixam em formatos específicos de objetos. No lado inferior direito está a imagem base para essa ilustração. Fonte Hui (2018)

Como o próprio nome sugere, Darknet-19 possui 19 camadas de convolução e outras 5 de *pooling* intercaladas como mostra a Tabela 2.1. Essa rede passa a prever 5 BBoxes para cada célula da matriz da imagem e é treinada para que, a cada 10 iterações, redimensione a imagem, num passo de 32 pixels, de 320x320 a 608x608 pixels, o que força a rede a aprender a prever objetos em entradas de diferentes dimensões (Redmon e Farhadi, 2017). A acurácia atingida no dataset Pascal VOC 2007 foi de 78.6%.

Para melhorar ainda mais o desempenho do sistema, Redmon e Farhadi (2018) introduziram a YOLOv3. Aqui é usado o mesmo princípio da YOLOv2 para a previsão de BBoxes com âncoras, porém, cada caixa pode agora ter mais de uma classe associada. Além disso, uma das maiores contribuições da YOLOv3 é a detecção feita em 3 escalas diferentes. A dimensão da imagem de entrada é reduzida em 32, 16 e 8 vezes para a detecção de objetos pequenos, médios e grandes, respectivamente. Por isso, a YOLOv3 tem melhor desempenho em relação às outras versões do sistema na detecção e classificação de objetos pequenos, embora perca precisão em relação aos objetos médios ou grandes (Redmon e Farhadi, 2018). A rede neuronal utilizada como *backbone* é a Darknet-53 com 53 camadas convolucionais, como mostra a Tabela 2.1.

Recentemente foi apresentado a YOLOv4, que tem como objetivo tornar possível um sistema de detecção de objetos em tempo real numa *Graphics Processing Unit* (GPU) convencional e, ao mesmo tempo, melhorar ainda mais a rápidez e precisão das versões anteriores (Bochkovskiy et al., 2020). Para isso, os autores propõem algumas otimizações para obter melhor precisão

Darknet-19

Tipo	Filtros	Tamanho	Saída
Convolucional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolucional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolucional	128	3×3	56×56
Convolucional	64	1×1	56×56
Convolucional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolucional	256	3×3	28×28
Convolucional	128	1×1	28×28
Convolucional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolucional	512	3×3	14×14
Convolucional	256	1×1	14×14
Convolucional	512	3×3	14×14
Convolucional	256	1×1	14×14
Convolucional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolucional	1024	3×3	7×7
Convolucional	512	1×1	7×7
Convolucional	1024	3×3	7×7
Convolucional	512	1×1	7×7
Convolucional	1024	3×3	7×7
Convolucional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Darknet-53

	Tipo	Filtros	Tamanho	Saída
	Convolucional	32	3×3	256×256
	Convolucional	64	$3 \times 3/2$	128×128
$\times 1$	Convolucional	32	1×1	
	Convolucional	64	3×3	
	Residual			128×128
	Convolucional	128	$3 \times 3/2$	64×64
$\times 2$	Convolucional	64	1×1	
	Convolucional	128	3×3	
	Residual			64×64
	Convolucional	256	$3 \times 3/2$	32×32
$\times 8$	Convolucional	128	1×1	
	Convolucional	256	3×3	
	Residual			32×32
	Convolucional	512	$3 \times 3/2$	16×16
$\times 8$	Convolucional	256	1×1	
	Convolucional	512	3×3	
	Residual			16×16
	Convolucional	1024	$3 \times 3/2$	8×8
$\times 4$	Convolucional	512	1×1	
	Convolucional	1024	3×3	
	Residual			8×8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Tabela 2.1: Arquiteturas Darknet-19 e Darknet-53 usadas na YOLOv2 e YOLOv3/v4, respectivamente. Fontes Redmon e Farhadi (2017) e Redmon e Farhadi (2018).

sem aumentar o custo computacional de inferência, como o uso de *data augmentation* e *class label smothing*. Também foram feitas otimizações que possuem um baixo custo computacional associado mas, em troca, aumentam significativamente a precisão do algoritmo. Exemplos dessas otimizações são o uso da função de ativação Mish e conexões extras nas camadas convolucionais para aumentar a capacidade de aprendizado da rede.

A estrutura da YOLOv4, ilustrada na Figura 2.10, está dividida em três blocos: *backbone*, *neck* e *head*. No primeiro bloco, onde são extraídas as características da imagem, YOLOv4 utiliza conexões *Cross-Stage-Partials* (CSP) (Wang et al., 2020a) entre as camadas da rede Darknet-53 (Tabela 2.1). Esse tipo de conexão é composto por blocos densos, baseados na DenseNet, onde cada camada está conectada a todas as anteriores (Figura 2.11). Os mapas de características obtidos são divididos em duas partes e, apenas uma delas, percorre todo o bloco denso para ser concatenada com a outra parte na próxima camada de transição (Hui, 2020), como mostra a Figura 2.12. Tal divisão dos mapas de características reduz a complexidade computacional do modelo de rede (Hui, 2020) por diminuir a quantidade de computações e parâmetros ajustáveis necessários.

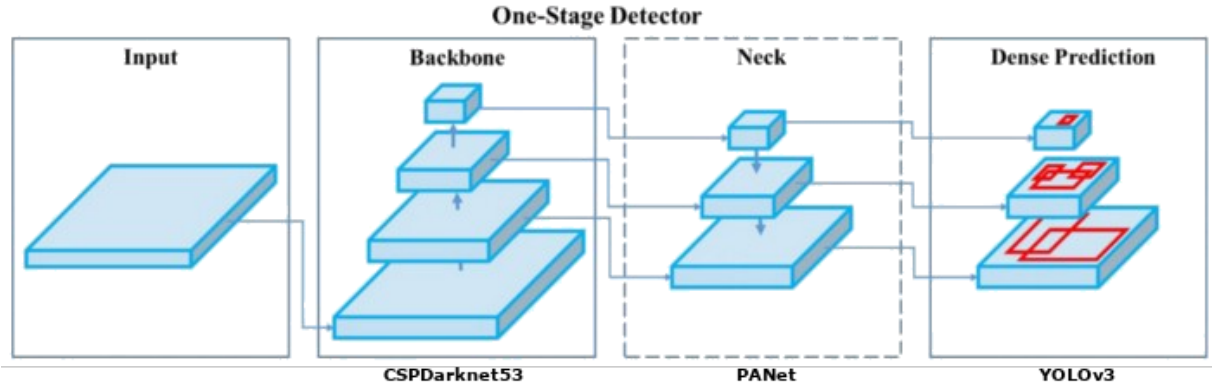


Figura 2.10: Estrutura da YOLOv4, detector de objetos de um estágio. Adaptado de Bochkovskiy et al. (2020).

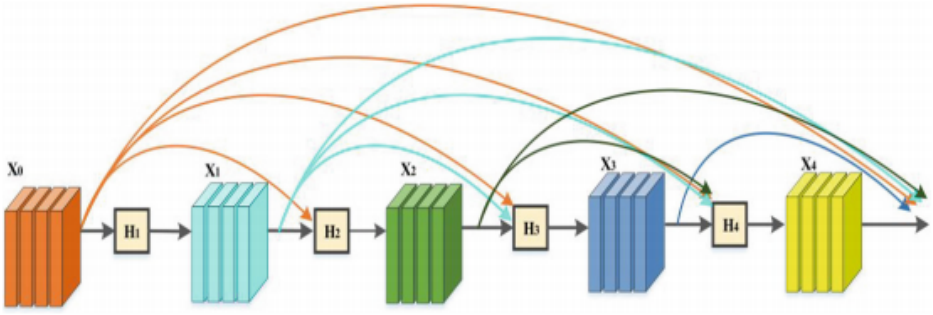


Figura 2.11: Um bloco denso de 5 camadas. Cada camada Hi é composta de *Batch Normalization*, ReLU e convolução. Hi recebe como entrada todos os mapas de características Xi das camadas anteriores, inclusive a entrada original. Fonte Wu e Tang (2021).

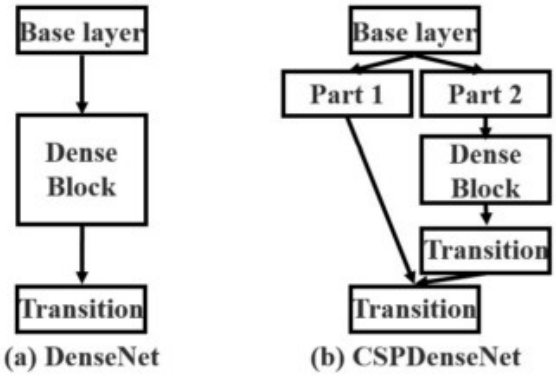


Figura 2.12: Em (b) CSPDenseNet, os mapas de características de entrada são divididos em duas partes, uma passa pelo bloco denso, baseado em (a) DenseNet, e é concatenada à outra parte para a camada de transição. Fonte Wang et al. (2020a).

Uma vez obtidos os mapas de características, estes são agrupados no estágio chamado *neck* usando a técnica *Path Aggregation Network* (PANet) (Liu et al., 2018). Por fim, a YOLOv3 é usada para detecção e classificação dos objetos, no estágio chamado de *head*.

YOLO continua avançando a cada ano e outras variantes também foram propostas, como a YOLOv3-Tiny (Adarsh et al., 2020), *PaddlePaddle-YOLO* (PP-YOLO) (Long et al., 2020), Scaled-YOLOv4 (Wang et al., 2021) e, ainda, YOLOv5 (Joseph Nelson, 2020). Todas têm

o objetivo em comum de aumentar a rapidez do sistema em *Frames* por segundo (FPS) para aplicações em tempo real e a comparação dos seus desempenhos estão ilustrados na Figura 2.13.

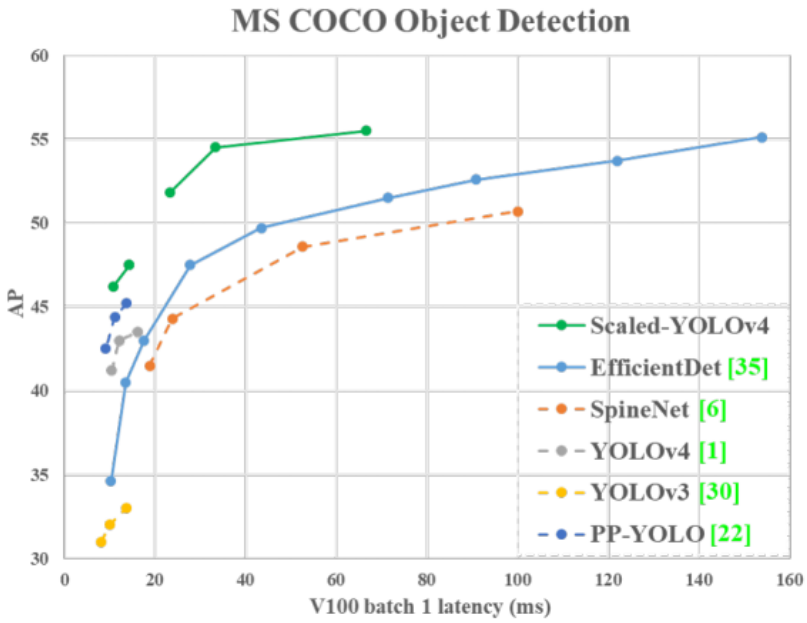


Figura 2.13: Comparação do desempenho no dataset COCO entre YOLOv3, YOLOv4, PP-YOLO e outros detectores de objetos. Fonte Wang et al. (2021).

2.3 MÉTRICAS DE AVALIAÇÃO

Para avaliar o desempenho do modelo em detectar objetos, as predições são dividas em 4 valores, cada um representando a quantidade de exemplares: Verdadeiro Positivo (TP), que quantifica as detecções feitas corretamente; Falso Positivo (FP), que representa detecções feitas incorretamente; Verdadeiro Negativo (TN), que representa a correta classificação de um objeto que não pertence à classe; e Falso Negativo (FN) que quantifica objetos de uma classe que não foram detectados, também chamado de *miss rate*. Essas taxas de erro (FP e FN) dependem principalmente da amostra de imagens selecionada para a fase de testes. Essa amostra pode não refletir a realidade de todo o *dataset* e, por isso, a taxa de FP pode ser maior. Por sua vez, FN é detectado quando o tamanho da amostra de imagens de teste é pequena ou tem alta variedade, o que não permite que a rede possa chegar a uma conclusão significativa sobre a representação que se quer detectar.

Como exemplo prático para entender esses conceitos, suponha que uma rede neuronal é capaz de diagnosticar doenças por imagem. Foram testados 15 pacientes e destes o teste identificou 10 como doentes. Entretanto, dos 10 identificados, apenas 8 estavam realmente doentes (TP), enquanto 2 estavam saudáveis (FP). E, 2 dos 5 que foram identificados inicialmente como saudáveis, possuiam a doença (FN), enquanto 3 foram identificados corretamente como saudáveis (TN). Nesse contexto, é extremamente prejudicial quando um paciente doente não é

identificado pela rede, por isso, é importante sempre reduzir casos de FP e FN e identificar casos relevantes (TP) corretamente. São diversas as métricas usadas para avaliar a performance de classificação de um detector. As usadas neste trabalho são a acurácia, precisão, precisão média e revocação.

Antes de introduzir as métricas citadas, é preciso definir outra necessária para validar uma detecção, a Interseção sobre União (IoU). Essa métrica avalia a sobreposição entre o valor ou BBox referência (do inglês, *Ground-Truth* (GT)) com a detecção feita. Para isso, é calculada a área da interseção dividido pela área da união das duas caixas, como ilustra a Figura 2.14. O valor resultante passa por um limiar para que o detector possa decidir se a detecção é valida ou não. O encaixe perfeito entre as duas caixas resulta em um $IoU = 1$, já $IoU = 0$ representa nenhum encaixe entre elas.

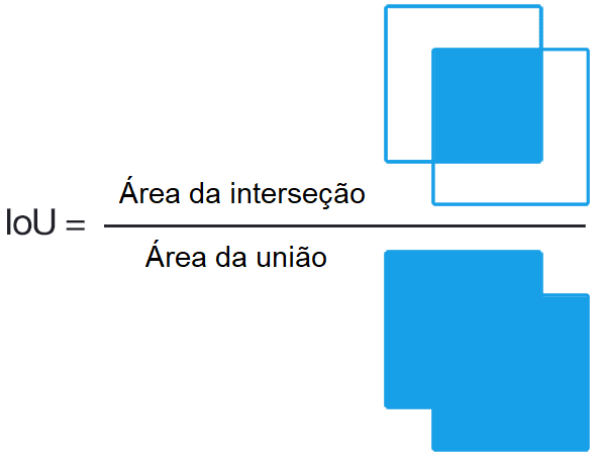


Figura 2.14: Interseção sobre União é igual a área da interseção dividido pela área da união das *bounding boxes* de referência e detectada. Adaptado de Rosebrock (2016).

A acurácia é definida como, o número de todas as detecções feitas corretamente dividido pelo total de detecções (Equação 2.1), ou seja, a fração de predições que foram feitas corretamente, independente da classe. Essa métrica não leva em conta classes separadas já que os resultados das detecções de todas as classes são condensadas. Como consequência disso, caso o número de imagens de treino e teste não estejam bem distribuídos entre as classes, a acurácia não é capaz de refletir o real desempenho da rede na detecção de objetos pertencentes às classes que carecem de exemplos.

$$\text{Acurácia} = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.1}$$

Por sua vez, revocação (do inglês, *recall*) mede a fração de detecções corretas entre todas as detecções que deveriam ter sido feitas (Equação 2.2), ou seja, dá a indicação da porcentagem

de objetos que foram perdidos na detecção de objetos de determinada classe. O valor resultante desta equação está entre 0 e 1, representando baixa e alta revocação, respectivamente.

$$\text{Revocação} = \frac{TP}{TP + FN} \tag{2.2}$$

Precisão é a porcentagem das detecções para determinada classe feitas corretamente (Equação 2.3). Essa métrica mede a consistência da rede em predizer corretamente os objetos. Precisão média (AP) e Média das precisões médias (mAP), do inglês, *mean Average Precision*, são as métricas mais conhecidas para avaliar detectores de objetos. Ambas resultam em valores entre 0 e 1, sendo que 0 representa baixa precisão e 1 alta precisão. Uma curva Precisão x Revocação (PR) é a representação da precisão em função da revocação do detector variando o seu valor de confiança ou limiar IoU. A área abaixo desta curva curva resulta no mAP do modelo.

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{2.3}$$

Outra métrica também conhecida no contexto de detectores de objetos é o *F-score*, também chamado de *F1-score* (Equacao 2.4). Essa métrica é uma maneira de encontrar um único valor médio harmônico entre precisão e revocação e usada para perceber o quão confiável está seu modelo em relação a um *dataset*. A média harmônica é usada para operandos inversamente proporcionais, ou seja, a medida que a precisão aumenta, a revocação tende a cair e vice-versa. O valor resultante, assim como os outros, está entre 0 e 1, sendo 1 o *F-score* do modelo ideal. Dependendo da aplicação, o número de FP e FN identificados possuem relevâncias diferentes e o *F-score* permite dar pesos diferentes entre essas duas medidas. Para o exemplo prático apresentado no início desta sessão, uma pessoa saudável detectada com a doença tem menor impacto do que uma pessoa doente identificada como saudável.

$$\text{F1-score} = 2 \times \frac{P \times R}{P + R} \tag{2.4}$$

3 TRABALHOS RELACIONADOS

A maioria dos trabalhos relacionados realizam a detecção dos olhos na face e estimação dos seus estados em etapas separadas. A detecção de face é um problema antigo que, mesmo que já hajam métodos consolidados e disponíveis em bibliotecas como o OpenCV, ainda se discute técnicas que sejam altamente eficazes e, ao mesmo tempo, rápidas para imagens ou vídeos em diferentes cenários. Alguns fatores influenciam na detecção da face como: a rotação e posição do rosto em relação à câmera; se há oclusão de parte do rosto por óculos, cabelos, maquiagem ou algum objeto; a expressão facial; se a imagem foi capturada em ambiente fechado ou aberto; se o fundo é complexo ou simples; as condições de luz, entre outros fatores.

Uma vez localizada a face e a área dos olhos, determinar o estado dos olhos como abertos ou fechados tem diversas aplicações. É usado, por exemplo, para detectar piscadas em um vídeo e com isso ajudar pessoas que não possuem movimentos dos membros a utilizarem computadores, detectar cansaço e fadiga de motoristas em trânsito, ou então detectar se uma pessoa está de olhos fechados numa foto. Este capítulo foi subdividido de modo a citar as técnicas utilizadas em cada uma das duas etapas introduzidas.

3.1 DETECÇÃO DE FACE E OLHOS

A grande maioria dos trabalhos até hoje utilizam o método proposto por Paul Viola e Michael Jones (Viola e Jones, 2001), que consiste em encontrar contrastes naturais da face por meio de filtros de Haar em classificadores combinados em uma estrutura em cascata - algoritmo chamado Adaboost (Lienhart e Maydt, 2002). Nesse estudo, os autores introduzem o conceito de imagem integral, onde cada pixel é a soma de todos os níveis de cinza dos pixels à esquerda e acima dele, exemplificada na Figura 3.1. Esse conceito facilita o cálculo de características da imagem e acelera o processo de detecção de faces nela.

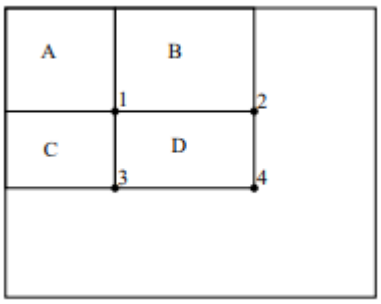


Figura 3.1: A soma dos pixels em uma região pode ser calculada com 4 referências. O valor da imagem integral na posição 1 é a soma dos pixels no retângulo A. O valor na posição 2 é A + B, na posição 3 é A + C e na posição 4 é A + B + C + D. A soma dos pixels da região D pode ser calculads como 4 + 1 - (2 + 3). Fonte (Viola e Jones, 2001).

Um método também muito utilizado na detecção de olhos é o de correspondência com o template do olho. Bhoi e Mohanty (2010) calculam essa correspondência com várias porções da imagem do rosto e aquela que resultar em maior valor é classificada como região do olho. Esse tipo de método geralmente tem uma boa acurácia mas são custosos computacionalmente e não tão robustos a diferentes cenários na captura da imagem (Tan et al., 2009), como variação na iluminação ou pose. Abordagens baseadas em características dos olhos detectam o centro e raio das pupilas, cantos dos olhos e o contorno das pálpebras. Diferente da maioria dos trabalhos, Zheng et al. (2005) utilizam imagens coloridas e usam a informação das cores para a detecção dessas características.

Em um trabalho mais recente, Wang et al. (2020b) propôs o modelo *Multitask Convolutional Neural Networks* (MTCNN) (Zhang et al., 2016) capaz de localizar múltiplas faces de forma rápida e precisa. Este modelo possui 3 estágios: o primeiro usa uma CNN pequena para rapidamente gerar uma série de janelas na imagem que possam conter um rosto; no segundo estágio usa uma CNN melhorada para filtrar a maioria das janelas que não possuem um rosto; no terceiro estágio, 5 pontos chaves do rosto são encontrados por uma outra CNN ainda melhor. Depois de encontrada a região da face e os pontos chaves, a imagem passa por uma série de transformações para localizar os olhos. São elas: transformação em escala de cinza, binarização e detecção da região dos olhos com base na projeção de gradiente vertical e horizontal. Uma revisão mais detalhada dessa área de pesquisa pode ser encontrada no trabalho de Song et al. (2013).

3.2 CLASSIFICAÇÃO DO ESTADO DO OLHO

Segundo Kim et al. (2017), os estudos para classificar o olho em aberto ou fechado podem ser separados em métodos não baseados em imagens, métodos baseados em vídeos, métodos baseados em imagens sem treinamento e com treinamento.

Nos métodos não baseados em imagens são analisadas atividades elétricas cerebrais captadas através de sensores conectados aos músculos dos olhos. Alguns exemplos são a eletro-oculografia, eletromiografia e eletroencefalograma. Embora sejam métodos rápidos, o usuário precisa conectar sensores ao seu corpo, o que limita seus movimentos. Métodos baseados em vídeos ou imagens não possuem tal limitação e têm a vantagem de obter informações através de imagens capturadas a longas distâncias (Kim et al., 2017).

Os estudos com vídeos detectam a piscada de olhos usando sucessivas imagens (i.e. *frames*). É uma técnica muito utilizada para a detecção de fadiga de motoristas no volante (Wang et al., 2020b) ou para auxiliar pessoas que não tem mobilidade a utilizar dispositivos eletrônicos (Mohammed e Anwer, 2014), por exemplo. Por ter de processar múltiplas imagens, esse tipo de método requer um tempo de processamento maior e, por isso, estudos ainda são feitos a fim de diminuir esse tempo e dar uma resposta em tempo hábil ao usuário.

Estudos baseados em imagens sem treinamento são aqueles que utilizam técnicas mais simples de processamento de imagens. Ji et al. (2018) determinam o estado do olho de acordo com a proporção entre o comprimento e a altura do menor retângulo externo ajustado no contorno do olho, obtido usando um algoritmo de extração da esclera. No estudo de Wang et al. (2017) é ajustado um círculo no contorno da pálpebra, obtida usando o filtro de Canny, e, a partir do ponto central e o raio deste círculo, é determinado o estado do olho. A partir de 6 pontos de interesse num dos olhos, obtidos através da técnica Intraface (Xiong e De la Torre, 2013), Soukupová e Čech (2016) calculam o *Eye Aspect Ratio* (EAR) usado para classificar o estado do olho. Porém, técnicas baseadas em pontos de interesse não são tão eficientes quando se trata de imagens em baixa resolução. Por isso, Mandal et al. (2017) propôs uma técnica que calcula a abertura dos olhos usando regressão linear e espectral da imagem da região do olho.

Como em geral a parte mais escura de uma imagem de olhos abertos binarizada é a iris, Mali e Lokhande (2014) encontraram uma forma de localizar o centro íris e a partir disto detectar o estado do olho. Este ponto é o que chamam de centro de gravidade - na física, ponto que equilibra todo o peso do corpo e, nesse estudo, é aquele concentra a maior parte dos pixels pretos da imagem. Em volta do ponto encontrado é recortado um quadrado de tamanho fixo e, considerando o número de pixels pretos dentro do quadrado e acima deste ponto, o olho é classificado em aberto ou fechado. A vantagem deste método em relação aos outros é que a imagem não precisa estar numa dimensão específica.

Como último exemplo de métodos baseados em imagens únicas sem treinamento, o trabalho de Gou et al. (2017) foi o primeiro a propôr realizar a detecção e estimação do grau de abertura dos olhos simultaneamente. No estudo é extraída a região dos olhos baseada nos 51 pontos de interesse da face obtidos por meio de (Wu e Ji, 2015). A partir da região extraída são considerados 5 pontos chaves nos olhos numa regressão linear em cascata que ajusta esses pontos a cada iteração. Esses pontos são inicializados em localizações médias do olho obtidas na fase de treinamento. Se, ao final da regressão, a probabilidade do olho estar aberto estiver abaixo de 20% então o olho é considerado fechado.

Técnicas com treinamento tem a vantagem de serem robustas a diferentes condições de luz, tamanho, cenário e rotação dos objetos. Lee et al. (2010) calculam a proporção da altura em relação à largura do olho e, juntamente com outros valores calculados, usam um *Support Vector Machine* (SVM) para a classificação. No estudo de Song et al. (2014) são analisados 3 tipos de características da imagem recortada e binarizada do olho: histogramas de gradientes orientados e variantes; padrões ternários locais, uma generalização dos padrões binários locais; e *Gabor Wavelets*. Essas características são aplicadas em três classificadores para determinar o estado do olho: *K-nearest neighbors*, SVM e Adaboost.

Yahyavi et al. (2016) extraem características (e.g. autovetores e autovalores) das imagens de cada olho que passam por uma rede neuronal *perceptron* multicamadas pré-treinada para classificar o estado do olho. Já Chirra et al. (2019) utilizam uma CNN para detectar o estado do olho e, no contexto de trânsito, classificar se o motorista está sonolento ou não. Kim et al. (2017)

comparam a performance de diversos modelos de redes neuronais pré-treinadas (ResNet50, AlexNet, VGG-16, GoogLeNet) com o classificador SVM, sendo a ResNet-50 adaptada a que obteve melhores resultados. No estudo de Wang e Qu (2021) são comparadas as redes neuronais Res-SE-net e AlexNet para determinar se o olho está aberto ou fechado. Yuan et al. (2021) utilizam o algoritmo YOLOv3-Tiny - versão simplificada do algoritmo YOLOv3 - para extrair faces humanas, olhos e boca da imagem. O estado do olho é depois determinado com a rede neuronal LeNet50.

As Tabelas 3.1 e 3.2 apresentam um resumo de todos os trabalhos relacionados estudados e as técnicas utilizadas em cada um, bem como os *datasets* mais utilizados na detecção de olhos e/ou seus estados.

Autor(es)	**Detecção do olho**	**Classificação em aberto ou fechado**	**Base(s) de Dados**	**# imagens/videos**
Wang e Qu (2021)	MTCNN	Res-SE-Net e AlexNet	Próprio, WIDERFACE, ZJU e YawDD	32.203 imagens
Yuan et al. (2021)	YOLOV3-Tiny	LeNet50	Próprio	8 vídeos
Wang et al. (2020b)	MTCNN	Processamento de imagens	Próprio	
Chirra et al. (2019)	AdaBoost	CNN	Próprio	2.850 imagens
Ji et al. (2018)	AdaBoost e EyeMap	Proporção	CIT, FERET e própria	1.361 imagens
Mandal et al. (2017)	OpenCV e SVM	Regressão linear	Próprio	
Gou et al. (2017)	Viola-Jones	Regressão linear	GI4E e BioID	
Wang et al. (2017)	AdaBoost	Contour Circle	Captação em tempo real	
Kim et al. (2017)	Não menciona	ResNet-50 CNN	Próprio e ZJU	610.050 imagens
Yahyavi et al. (2016)	AdaBoost	MLP	Própria	640 imagens
Soukupová e Čech (2016)	Intraface e Chehra	EAR e SVM	300-VW e ZJU	130 vídeos
Mali e Lokhande (2014)	Centro de gravidade	Observações	Próprio	543 imagens
Song et al. (2014)	Viola-Jones + HOG, LTP e Gabor	KNN, SVM e Adaboost	ZJU e CEW	9.757 imagens
Lee et al. (2010)	AdaBoost e LKT	SVM	Próprio, ZJU e TFV [1]	86 vídeos

Tabela 3.1: Trabalhos relacionados à classificação do estado do olho e os métodos utilizados por cada um para a detecção da região dos olhos. A classificação pela proporção e regressões lineares é feita em relação ao grau de abertura dos olhos. Kim et al. (2017) utilizaram *data augmentation* para obter as 610.050 imagens finais.

Nome	**Tamanho**	**Anotações**	**Ambiente**	**#faces/imagem**
WIDERFACE	32.203 imagens	Posição da face	Não controlado	12
FERET	14.126 imagens	Não especificado	Semi-controlado	1
300-VW	114 vídeos	68 pontos na face	Não controlado	1
300-W	399 imagens	68 pontos na face	Não controlado	1.5
TFV	5000 imagens	68 pontos na face	Semi-controlado	1
ZJU	80 vídeos	Não especificado	Controlado	1
GI4E	1339 imagens	Posição dos olhos	Semi-controlado	1
BioID	1521 imagens	Posição dos olhos	Não controlado	1
CEW	2423 imagens	Posição dos olhos abertos e fechados	Não controlado	1

Tabela 3.2: Base de dados utilizadas nos estudos relacionados. Com excessão do *dataset* FERET, todos estão publicamente disponibilizados. A quantidade de faces por imagem é uma média de todas as imagens.

[1] TFV: Talking Face Video

4 MATERIAIS E MÉTODOS

Neste capítulo serão descritos com detalhes como o estudo e experimentos foram conduzidos. A primeira seção dá mais informações sobre o dataset utilizado e os procedimentos para obtê-lo. Depois, são descritos como foram feitos os experimentos e ferramentas e tecnologias envolvidas. Por fim, a última seção apresenta como os resultados foram analisados e quais as métricas utilizadas para isso.

Todas as imagens e anotações da região dos olhos e seu estado utilizados neste trabalho, bem como sua separação em conjuntos de treinamento, teste e validação, estão disponíveis no GitHub[1].

4.1 DATASET

Como o objeto de estudo deste trabalho envolve fotografias de uma ou mais pessoas em ambiente não controlados, se fez necessária a escolha de um *dataset* com imagens variadas. Os mais utilizados nos trabalhos relacionados, apresentados no Capítulo 3, possuem diversas características diferentes, mas uma em comum, cada imagem possui, em média, apenas um rosto e/ou um par de olhos anotados. Foi escolhido, portanto, o *dataset* WIDERFACE, disponível publicamente, como base para a seleção de imagens e criação de anotações do estado dos olhos. Esse *dataset* é originalmente usado para avaliar detectores de face e possui imagens de diferentes resoluções, grande variedade de indivíduos, escala, pose, expressões, maquiagens, iluminação e fundo, conforme mostra a Figura 4.1.

A base de imagens WIDERFACE possui 32.203 exemplares retirados da *internet* organizados em 61 eventos e, para cada evento, os autores selecionaram 40% para o conjunto de treinamento, 10% para validação e 50% para teste. Ao todo são 393,703 faces anotadas com coordenadas das BBoxes e indicadores de expressão típica ou exagerada, iluminação normal ou extrema, oclusão parcial ou total da face, pose típica ou atípica e se a face é válida ou não (Yang et al., 2016).

Neste trabalho foi utilizado o conjunto de 10% do total de imagens que os autores propõem para a validação. Das 3.226 imagens e 39.708 faces anotadas deste conjunto, 585 são faces inválidas, ou seja, possuem menos de 10 pixels ou não são reconhecidas por baixa resolução da imagem, 31.996 são faces pequenas (10-50 pixels), 6.693 são médias (50-300 pixels) e 434 grandes (mais de 300 pixels). Para definir uma altura mínima da face tal que o estado dos olhos seja detectável, analisou-se imagens com faces de 10 a 40 pixels, ilustradas na figura 4.2. Foram pré-selecionadas, portanto, imagens que possuem faces válidas e que tenham pelo menos 40 pixels de altura - o mínimo estabelecido - totalizando, 1968 imagens selecionadas e 4663

[1]GitHub: https://github.com/lauraslopes/WIDERFACE-EyesAnnotation

Figura 4.1: Exemplos de imagens presentes no WIDERFACE e as *bounding boxes* em verde nas faces anotadas. O *dataset* possui uma grande variedade em escala, pose, oclusão, expressão e iluminação nas imagens. Retirado e traduzido de Yang et al. (2016).

faces anotadas que foram usadas como base para este trabalho. Essas imagens foram divididas aleatoriamente entre os conjuntos de treinamento (40%), teste (20%) e validação (40%).

Como nas anotações originais não há indicação da posição dos olhos, foi preciso anotá-los para que pudessem ser usados no treinamento e testes dos classificadores. As anotações foram feitas através da ferramenta *Computer Vision Annotation Tool* (CVAT)[2] que fornece instrumentos para rotular objetos em imagens e vídeos em formatos diversos: caixas, polígonos, polilinhas e pontos. A Figura 4.3 apresenta uma imagem presente no *dataset* e que foi anotada usando a ferramenta. As BBoxes contém pelo menos um olho humano, pálpebra inferior e superior e a política adotada para determinar o estado dos olhos foi a seguinte:

[2]CVAT: https://openvinotoolkit.github.io/cvat/about/. Último acesso em 07/12/2021.

Figura 4.2: Exemplos de imagens que contém faces entre 10 e 40 pixels de altura, presentes no *dataset* WIDERFACE. Analisou-se que não é possível determinar o estado dos olhos de faces com menos de 40 pixels de altura.

- Os olhos são considerados **abertos** se a esclera e a pupila (ponto central da íris) estão visíveis;
- Caso contrário e/ou o estado for indeterminado, os olhos são considerados **fechados**.

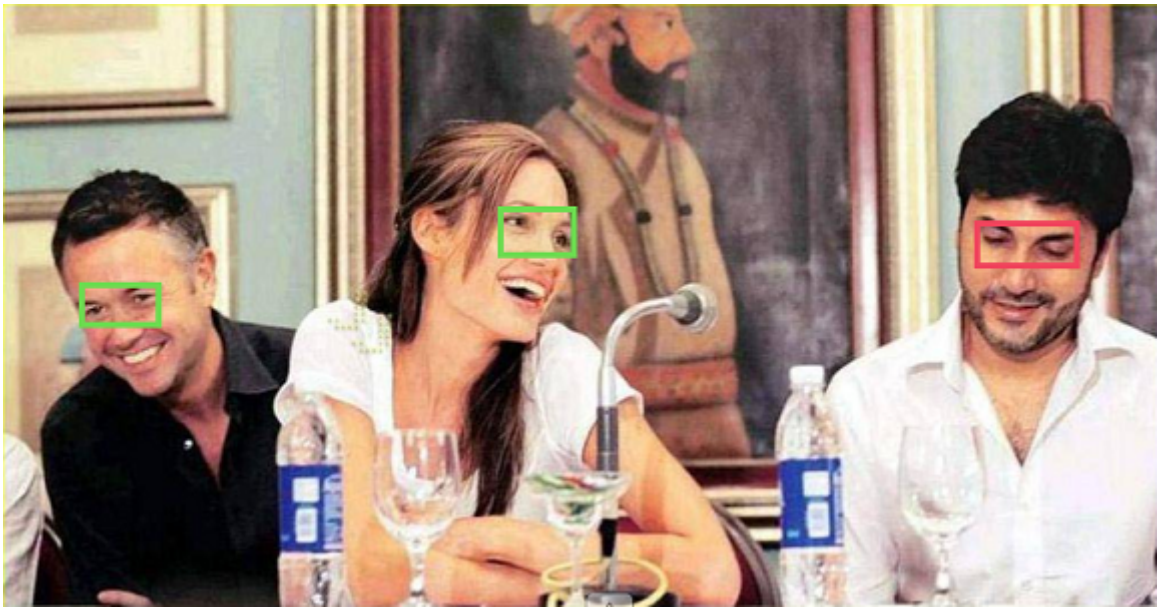


Figura 4.3: Exemplo de imagem presente no *dataset* com as anotações dos olhos feitas através da ferramenta CVAT. As *bounding boxes* verdes correspondem a olhos abertos e as vermelhas correspondem a olhos fechados.

As anotações foram exportadas no formato utilizado pela YOLO que possui o identificador da classe a qual os olhos pertencem - aberto ou fechado - e coordenadas (x, y) centrais e altura e largura da BBox. As coordenadas têm valores entre 0 e 1 relativas ao tamanho original da imagem (Bochkovskiy, 2021), por exemplo, $altura\_relativa = altura\_BBox / altura\_imagem$.

Ao todo foram 3.975 olhos anotados dos quais 2.890 estão abertos e 1.085 estão fechados. A quantidade de olhos anotados é inferior ao de faces pois os autores do *dataset* original consideraram face válida a que contém, pelo menos, testa, queixo e bochecha (Yang et al., 2016) e, portanto, os olhos não necessariamente estão visíveis.

Imagens podem conter tanto olhos fechados quanto abertos, como também ilustra a Figura 4.3, e, cerca de 33% delas possuem olhos fechados, o que equivale a 649 imagens, e 76%

possuem olhos abertos, equivalente à 1.492 imagens. As BBoxes anotadas têm um tamanho ($altura \times largura$) médio de 1.309 pixels e desvio padrão de 2.939. Foram anotados olhos de diversos tamanhos, pequenos, médios e grandes, sendo que, 3.115 estão abaixo da média de tamanho e 859 acima.

4.2 EXPERIMENTOS

Todos os experimentos foram realizados em uma máquina com processador Quad-Core AMD Opteron(tm) 8387 2.8GHz com 64GB de RAM e GPU NVIDIA Titan X com 12GB de RAM e 3.072 núcleos CUDA. CUDA versão 10.0, cuDNN 6, Tensorflow 2.1.0 e Keras 2.2.1.

A primeira parte dos experimentos foi avaliar o desempenho da detecção e classificação do estado dos olhos em apenas uma rede neuronal, o que se chamou de classificação em uma etapa. O modelo escolhido para essa tarefa foi a YOLOv4, descrita no Capítulo 2, por ser capaz de detectar e classificar objetos de diversas classes em apenas uma varredura na imagem.

Para o treinamento foram utilizados parâmetros pré-treinados fornecidos por Bochkovskiy (2021) para a YOLOv4. A configuração da rede possui dimensão de entrada de 608x608 pixels, portanto, todas as imagens e anotações GT são redimensionadas antes de serem usadas, e as coordenadas das predições feitas são relativas a esse novo tamanho. O número de iterações (*batches*) na fase de treinamento foi de 10.000, com ajustes na taxa de aprendizagem (*steps*) nas iterações 8.000 e 9.000. Foi ativada a opção *random* de redimensionar as imagens para diferentes tamanhos para contribuir com a generalização da escala dos objetos. Por fim, o limiar IoU foi definido como 0.7.

A segunda parte dos experimentos teve como objetivo avaliar o desempenho em detectar e classificar o estado dos olhos em redes neuronais diferentes, o que se chamou de classificação em duas etapas. Foi utilizada novamente a YOLOv4 para a primeira etapa de detecção da região dos olhos a partir do treinamento com o mesmo *dataset* descrito na seção anterior, porém, considerando as duas classes - aberto e fechado - como apenas uma. A configuração da rede tem a mesma dimensão de entrada do experimento anterior (608x608 pixels), limiar IoU de 0.7 e opção *random* ativada, entretanto possui 6.000 iterações e ajustes na taxa de aprendizagem nas iterações 4.800 e 5.400.

Para a etapa de classificação do estado dos olhos foi usada uma ResNet-50, também descrita no Capítulo 2, implementada usando *Tensorflow* e *Keras*, bibliotecas que facilitam a criação e treinamento de redes neuronais. A base de imagens usada aqui origina do recorte das imagens usando as anotações dos olhos feitas e descritas na seção anterior. Os recortes foram separados em dois conjuntos, treinamento (80%, imagens originadas dos conjuntos de treinamento e validação previamente separados) e teste (20%). A rede foi configurada com uma dimensão de entrada de 224x224 pixels, função de ativação *softmax* e parâmetros pré-treinados com o *dataset* ImageNet.

4.3 ANÁLISE DOS DADOS

Os resultados foram analisados utilizando as métricas de precisão, revocação, e *F1-Score* conforme descritos no Capítulo 2. Para obter estes valores o limiar de confiança foi variado de [0 a 1), no passo de 0.1 a cada teste. O objeto detectado que possuir porcentagem de confiança acima deste limiar é classificado pela rede e, com os valores de TP, FP e FN as métricas são calculadas. Além da precisão e revocação média das redes, também foi calculado estes valores para cada classe de estado dos olhos. Também foi feita uma avaliação visual comparando as detecções obtidas nos experimentos e os valores das BBoxes de referência (GT). Os resultados obtidos pelos experimentos são descritos no capítulo a seguir.

5 RESULTADOS

Este trabalho teve como objetivo principal verificar o desempenho da detecção e classificação simultânea do estado dos olhos em fotografias capturadas em ambientes não controlados. Para isso, foi avaliada a rede neuronal YOLOv4 aplicada a este cenário e uma combinação entre YOLOv4 e ResNet-50 para avaliar o desempenho da detecção e classificação em etapas diferentes. Espera-se que os resultados obtidos em apenas uma etapa sejam semelhantes ou superiores a da técnica com duas etapas.

O primeiro desafio da fase de experimentação foi encontrar a base de imagens mais condizente com o que se pretendia verificar, ou seja, que tenham imagens com grande variedade de número de indivíduos, pose, expressões, fundo, resolução, iluminação, entre outras variáveis. A grande maioria dos *datasets* disponíveis para a classificação do estado dos olhos, i.e. GI4E e CEW, possuem apenas uma face por imagem e, portanto, não se aplicam a este estudo. Por isso, foi escolhido o *dataset* WIDERFACE que atende todas as características desejadas nas imagens. O conjunto de imagens passou por uma filtragem para desconsiderar aquelas que possuiam faces inválidas ou menores que 40 pixels e, por fim, foi feita uma nova anotação da região dos olhos usando a ferramenta CVAT, como detalhado no Capítulo 4.

A cada etapa de teste dos modelos de rede variou-se o limiar de confiança, que determina se o objeto foi detectado ou não, de 0 a 1. Os valores obtidos de precisão e revocação foram reunidos para construir as curvas apresentadas na Figura 5.1 e calcular o *F-Score* máximo para cada classe. As curvas pontilhadas foram obtidas nos testes da classificação em duas etapas (YOLOv4 + ResNet-50) e as curvas preenchidas dizem respeito aos testes da classificação em apenas uma etapa utilizando YOLOv4.

Nos experimentos de apenas uma etapa, o *F-Score* máximo obtido para a classificação de olhos fechados foi de 0.42 e para olhos abertos foi de 0.80. Já nos experimentos de duas etapas, o *F-Score* máximo para a classificação de olhos fechados pela ResNet-50 foi de 0.60, 42% a mais que o obtido pelo experimento anterior. E, para a classificação de olhos abertos, foi de 0.82, 2.5% a mais que o obtido pela YOLOv4.

A precisão média obtida pela YOLOv4 no primeiro experimento foi de 82.58% para olhos abertos e 35.92% para olhos fechados, resultando em uma média das precisões das duas classes (mAP) de 59.25%. No experimento em duas etapas, por sua vez, a YOLO que detecta apenas 1 classe obteve a precisão média de 87.79% para olhos e a ResNet-50 teve precisão de 55% para olhos fechados e 86% para olhos abertos, resultando em mAP de 70.5%.

Uma vez obtidos os dados e analisando o gráfico das curvas de precisão x revocação, é possível perceber que o experimento em duas etapas teve melhor desempenho em classificar, principalmente, olhos fechados, diferente do que era esperado desta pesquisa. Embora, identificar olhos fechados ainda não teve desempenho tão bom quanto a identificação dos olhos abertos. O

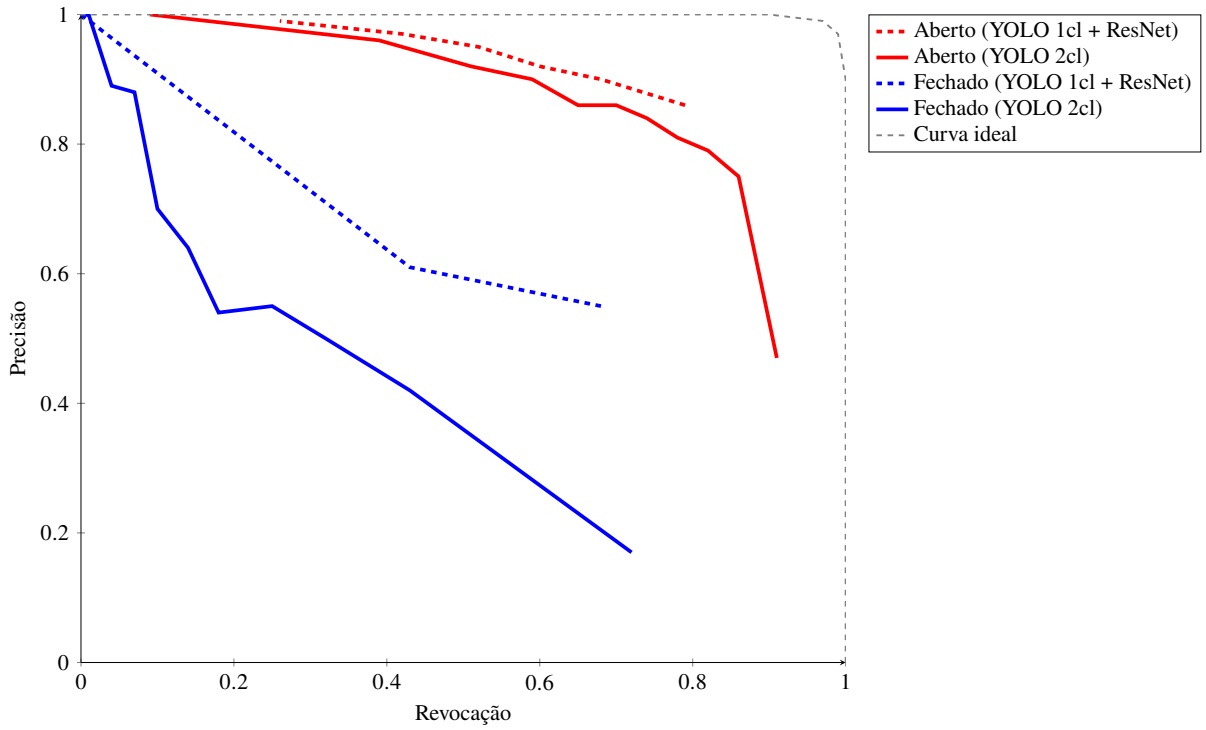


Figura 5.1: Curvas precisão x revocação da classificação do estado dos olhos. As curvas pontilhadas são relativas aos experimentos da classificação pela ResNet-50 após a detecção da região dos olhos (1 classe) feita pela YOLOv4. Já as preenchidas dizem respeito à detecção e classificação das duas classes feitas simultaneamente pela YOLOv4.

que as curvas apresentadas nos mostram é que, a medida que o limiar de confiança aumenta, mais olhos de determinada classe são detectados, ou seja, menos os casos de FN e maior a revocação. Porém, isso implica que mais casos de FP também são identificados e, portanto, menor fica a precisão. A curva ideal possui precisão, revocação e *F-Score* iguais a 1, o que fica muito distante das obtidas pelas classificações.

Para a classificação de olhos fechados nos dois experimentos, a revocação não ultrapassa 72%, o que significa que pelo menos 28% dos olhos fechados não são detectados corretamente. Uma possível maneira de contornar esse problema, é fornecer para a rede mais exemplares de olhos fechados utilizando técnicas de *data augmentation*, por exemplo. Assim, o modelo será capaz de generalizar melhor a representação e obter melhores resultados.

5.1 AVALIAÇÃO VISUAL

Embora a área de detecção de objetos tenha avançado muito nos últimos anos, ainda há desafios a se enfrentar. Um deles é contruir um detector capaz de aprender de forma genérica e seja robusto as mais diversas variações possíveis de um mesmo objeto, ambiente ou características da imagem. Além da avaliação das métricas, uma avaliação visual é útil para que seja possível perceber em que situações o detector falha e hipotetizar motivos e soluções para corrigir os erros.

Nesta seção são avaliados e comparados exemplos de predições feitas pela YOLOv4 ao detectar e classificar simultaneamente olhos abertos e fechados em fotografias em ambientes não controlados.

5.1.1 Falsos Negativos

Um falso negativo é identificado quando o objeto de uma classe não foi detectado pelo modelo de rede. Na Figura 5.2 A, alguns olhos deixaram de ser classificados pela presença de maquiagem marcante na face. O modelo também deixa de detectar olhos das duas classes em fotografias com parte da região dos olhos oclusa por sombra causada por má iluminação do ambiente (Figura 5.2 B).

Figura 5.2: Imagens onde não foram encontrados olhos de nenhuma das classes pela maquiagem e sombra na região dos olhos. As caixas delimitadoras verdes correspondem a olhos abertos e as caixas vermelhas correspondem a olhos fechados.

A Figura 5.3 apresenta exemplos de predições de falsos negativos em fotografias onde há oclusão parcial da face por uso de capacete, máscaras e outros acessórios (A) ou oclusão da região dos olhos pelo uso de óculos (B).

Figura 5.3: Imagens onde não foram encontrados olhos de nenhuma das classes por oclusão parcial da face ou região dos olhos.As caixas delimitadoras verdes correspondem a olhos abertos e as caixas vermelhas correspondem a olhos fechados.

O detector também apresenta dificuldade em detectar a região dos olhos e classificar o seu estado em faces que estão rotacionadas ou em pose não habitual, por exemplo quando o rosto está de lado e apenas um dos olhos está visível. Exemplos são apresentados na Figura 5.4.

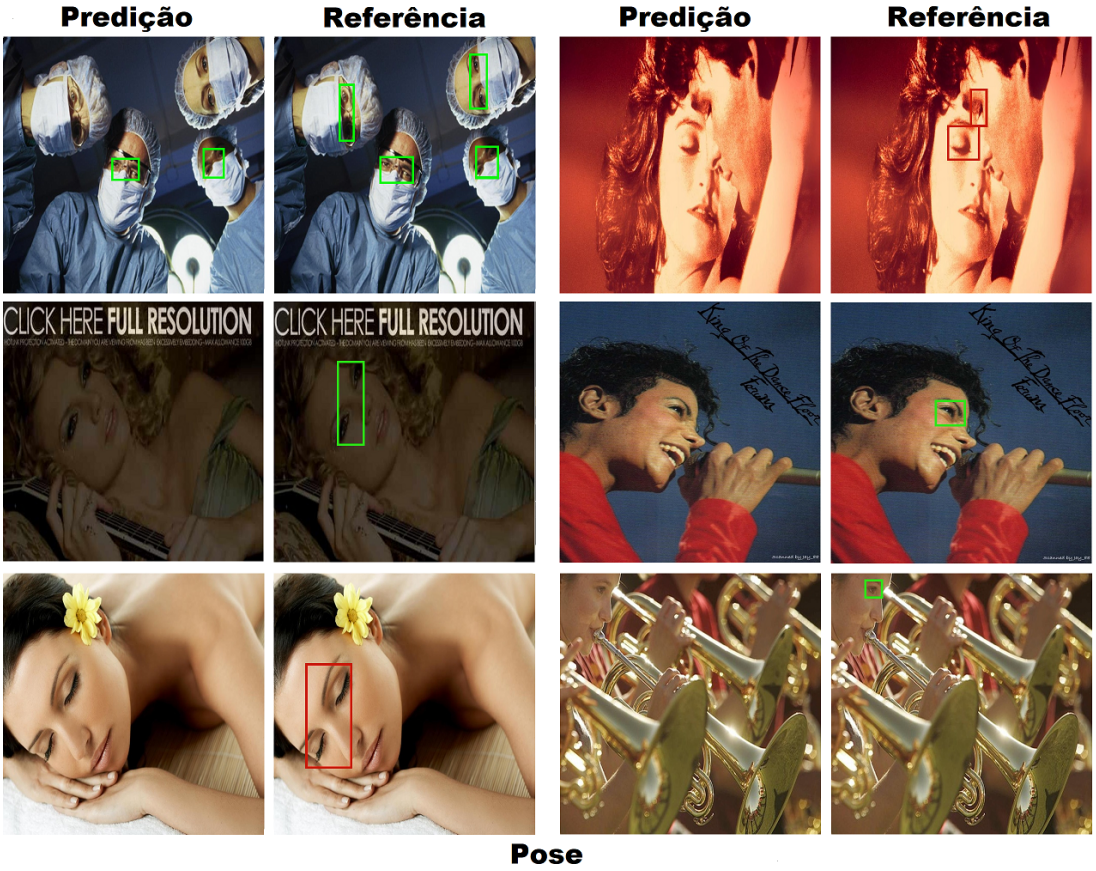


Figura 5.4: Imagens onde não foram encontrados olhos de nenhuma das classes por pose não habitual e rotação da face. As caixas delimitadoras verdes correspondem a olhos abertos e as caixas vermelhas correspondem a olhos fechados.

5.1.2 Falsos Positivos

São considerados falsos positivos as detecções de qualquer estado em lugares onde não há olhos e a classificação cruzada de olhos abertos para olhos fechados e vice-versa. A Figura 5.5(A) apresenta alguns exemplos de classificações cruzadas que, nos casos apresentados, podem ter acontecido pelas poses ou nitidez das fotos. Já na Figura 5.5 B, estão exemplos de predições de olhos em qualquer estado onde sequer haviam olhos.

Figura 5.5: A) Exemplos de imagens com classificação cruzada - onde foram encontrados olhos fechados mas as predições corretas eram abertos e vice-versa. B) Exemplos de imagens onde foram identificados olhos onde não haviam. As caixas delimitadoras verdes correspondem a olhos abertos e as caixas vermelhas correspondem a olhos fechados.

5.1.3 Verdadeiros Positivos e Negativos

Verdadeiros positivos e negativos ocorrem quando são feitas predições corretas tanto para a presença de objetos da(s) classe(s) ou não. Na Figura 5.6 é possível ver alguns exemplos de TP e TN identificados em fotografias diversas, com uma ou mais pessoas presentes, diferentes fundos, expressões, cores, poses e escalas.

Figura 5.6: Exemplos de imagens que tiveram os olhos - e a ausência de olhos - identificados corretamente. As caixas delimitadoras verdes correspondem a olhos abertos e as caixas vermelhas correspondem a olhos fechados.

6 CONCLUSÃO

Buscou-se, com esse trabalho, apresentar técnicas de *deep learning* para classificar o estado dos olhos em fotografias capturadas em ambientes não controlados. Para construir o *dataset* utilizado nos experimentos, foi utilizada a ferramenta CVAT para anotar olhos fechados e abertos em imagens selecionadas do *dataset* WIDERFACE. Ao todo, foram 1.968 imagens e 3.975 olhos anotados e estão disponíveis no GitHub `https://github.com/lauraslopes/WIDERFACE-EyesAnnotation`.

No primeiro experimento, foi utilizada a rede neuronal YOLOv4 para a detecção e classificação simultânea do estado dos olhos. O segundo experimento utilizou também a YOLOv4 para a detecção de olhos nas imagens e, em seguida, a ResNet-50 para a classificação dos seus estados.

Os resultados obtidos indicam que realizar a classificação em duas etapas diferentes tem melhor desempenho que realizar as etapas simultaneamente em apenas uma rede neuronal. Além disso, com uma avaliação visual foi possível identificar os casos em que a YOLOv4 mais falha em classificar corretamente o estado dos olhos. Por exemplo, em imagens com má iluminação ou baixa resolução, indivíduos em poses não habituais, com maquiagem marcante e ou que estejam com a face oclusa parcialmente.

Porém, uma melhor investigação é necessária para identificar maiores motivos para a diferença do desempenho entre os dois modelos utilizados. Além de uma investigação do quão mais rápida e eficaz computacionalmente foi realizar a detecção e classificação do estado dos olhos simultaneamente.

Pretende-se também, no futuro, utilizar técnicas de *data augmentation* para aumentar a quantidade de imagens e variabilidade do *dataset* utilizado, a fim de diminuir os casos de falha dos modelos. Além de estudar e comparar resultados de outras redes neuronais para a mesma aplicação como a *EfficientDet* e *Single Shot MultiBox Detector* (SSD).

REFERÊNCIAS

Adarsh, P., Rathi, P. e Kumar, M. (2020). Yolo v3-tiny: Object detection and recognition using one stage improved model. Em *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, páginas 687–694.

Bhoi, N. e Mohanty, M. (2010). Template matching based eye detection in facial image. *International Journal of Computer Applications*, 12.

Bochkovskiy, A. (2021). Yolo v4, v3 and v2 for windows and linux. Último acesso em 07/12/2021, `https://github.com/AlexeyAB/darknet`.

Bochkovskiy, A., Wang, C. e Liao, H. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934.

Boesch, G. (2021). Deep residual networks (resnet, resnet50) – guide in 2021. Último acesso em 02/12/2021, `https://viso.ai/deep-learning/resnet-residual-neural-network/`.

Chirra, V., Reddy, U. S. e KishoreKolli, V. (2019). Deep cnn: A machine learning approach for driver drowsiness detection based on eye state. *Revue d'Intelligence Artificielle*, 33:461–466.

Dettmers, T. (2015). Understanding convolution in deep learning. Último acesso em 10/06/2021, `https://timdettmers.com/2015/03/26/convolution-deep-learning/`.

Goodfellow, I., Bengio, Y. e Courville, A. (2016). *Deep Learning*. MIT Press. Último acesso em 16/11/2021, `http://www.deeplearningbook.org`.

Gou, C., Wu, Y., Wang, K., Wang, K., Wang, F.-Y. e Ji, Q. (2017). A joint cascaded framework for simultaneous eye detection and eye state estimation. *Pattern Recognition*, 67:23–31.

He, K., Zhang, X., Ren, S. e Sun, J. (2016). Deep residual learning for image recognition. Em *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 770–778.

Hui, J. (2018). Real-time object detection with yolo, yolov2 and now yolov3. Último acesso em 27/06/2021, `https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088`.

Hui, J. (2020). Yolov4. Último acesso em 10/07/2021, `https://jonathan-hui.medium.com/yolov4-c9901eaa8e61`.

Ji, Y., Wang, S., Lu, Y., Wei, J. e Zhao, Y. (2018). Eye and mouth state detection algorithm based on contour feature extraction. *Journal of Electronic Imaging*, 27:1.

Joseph Nelson, J. S. (2020). Yolov5 is here: State-of-the-art object detection at 140 fps. Último acesso em 12/07/2021, `https://blog.roboflow.com/yolov5-is-here/`.

Kim, K. W., Hong, H. G., Nam, G. P. e Park, K. R. (2017). A study of deep cnn-based classification of open and closed eyes using a visible light camera sensor. *Sensors*, 17(7).

LeCun, Y., Bengio, Y. e Hinton, G. (2015). Deep learning. *Nature*, 521:436–44.

Lee, W. O., Lee, E. C. e Park, K. R. (2010). Blink detection robust to various facial poses. *Journal of Neuroscience Methods*, 193(2):356–372.

Lienhart, R. e Maydt, J. (2002). An extended set of haar-like features for rapid object detection. Em *Proceedings of the International Conference on Image Processing*, volume 1, páginas I–900.

Lin, T., Dollár, P., Girshick, R. B., He, K., Hariharan, B. e Belongie, S. J. (2016). Feature pyramid networks for object detection. *CoRR*, abs/1612.03144.

Liu, S., Qi, L., Qin, H., Shi, J. e Jia, J. (2018). Path aggregation network for instance segmentation. Em *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, páginas 8759–8768.

Long, X., Deng, K., Wang, G., Zhang, Y., Dang, Q., Gao, Y., Shen, H., Ren, J., Han, S., Ding, E. e Wen, S. (2020). Pp-yolo: An effective and efficient implementation of object detector.

Mali, H. B. e Lokhande, S. D. (2014). Eye state detection using center of gravity approach. Em *2014 Annual IEEE India Conference (INDICON)*, páginas 1–4.

Mandal, B., Li, L., Wang, G. S. e Lin, J. (2017). Towards detection of bus driver fatigue based on robust visual analysis of eye state. *IEEE Transactions on Intelligent Transportation Systems*, 18(3):545–557.

Michaelis (2015). Dicionário brasileiro da língua portuguesa. Último acesso em 10/06/2021, `https://michaelis.uol.com.br/moderno-portugues/busca/portugues-brasileiro/representa%C3%A7%C3%A3o/`.

Mohammed, A. P. A. A. e Anwer, M. S. S. A. (2014). Efficient eye blink detection method for disabled-helping domain. *International Journal of Advanced Computer Science and Applications*, 5(5).

Ponti, M. A., Ribeiro, L. S. F., Nazare, T. S., Bui, T. e Collomosse, J. (2017). Everything you wanted to know about deep learning for computer vision but were afraid to ask. Em *2017 30th*

SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T), páginas 17–41.

Redmon, J., Divvala, S., Girshick, R. e Farhadi, A. (2016). You only look once: Unified, real-time object detection. Em *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 779–788.

Redmon, J. e Farhadi, A. (2017). Yolo9000: Better, faster, stronger. Em *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 6517–6525.

Redmon, J. e Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv*.

Rosebrock, A. (2016). Intersection over union (iou) for object detection. Último acesso em 16/11/2021, `https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/`.

Song, F., Tan, X., Chen, S. e Zhou, Z.-H. (2013). A literature survey on robust and efficient eye localization in real-life scenarios. *Pattern Recognition*, 46(12):3157–3173.

Song, F., Tan, X., Liu, X. e Chen, S. (2014). Eyes closeness detection from still images with multi-scale histograms of principal oriented gradients. *Pattern Recognition*, 47(9):2825–2838.

Soukupová, T. e Čech, J. (2016). Real-time eye blink detection using facial landmarks. Em *21st Computer Vision Winter Workshop (CVWW)*.

Tan, X., Song, F., Zhou, Z.-H. e Chen, S. (2009). Enhanced pictorial structures for precise eye localization under incontrolled conditions. Em *2009 IEEE Conference on Computer Vision and Pattern Recognition*, páginas 1621–1628.

Tejani, S. (2016). Machines that can see: Convolutional neural networks. Último acesso em 10/06/2021, `https://shafeentejani.github.io/2016-12-20/convolutional-neural-nets/`.

Viola, P. e Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Em *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, páginas I–I.

Voinov, S. (2020). *Deep Learning-based Vessel Detection from Very High and Medium Resolution Optical Satellite Images as Component of Maritime Surveillance Systems*. Tese de doutorado, Universität Rostock.

Wang, C.-Y., Bochkovskiy, A. e Liao, H.-Y. M. (2021). Scaled-yolov4: Scaling cross stage partial network.

Wang, C.-Y., Mark Liao, H.-Y., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W. e Yeh, I.-H. (2020a). Cspnet: A new backbone that can enhance learning capability of cnn. Em *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, páginas 1571–1580.

Wang, M., Guo, L. e Chen, W.-Y. (2017). Blink detection using adaboost and contour circle for fatigue recognition. *Computers & Electrical Engineering*, 58:502–512.

Wang, Y. e Qu, R. (2021). Research on driver fatigue state detection method based on deep learning. *Journal of Physics: Conference Series*, 1744(4):042242.

Wang, Z., Shi, P. e Wu, C. (2020b). A fatigue driving detection method based on deep learning and image processing. *Journal of Physics: Conference Series*, 1575:012035.

Wu, X. e Tang, R. (2021). Fast detection of passion fruit with multi-class based on yolov3. Em Jia, Y., Zhang, W. e Fu, Y., editores, *Proceedings of 2020 Chinese Intelligent Systems Conference*, páginas 818–825, Singapore. Springer Singapore.

Wu, Y. e Ji, Q. (2015). Shape augmented regression method for face alignment. Em *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, páginas 979–985.

Xiong, X. e De la Torre, F. (2013). Supervised descent method and its applications to face alignment. Em *2013 IEEE Conference on Computer Vision and Pattern Recognition*, páginas 532–539.

Yahyavi, S., Mazinan, A. e Khademi, M. (2016). Real-time high-resolution detection approach considering eyes and its states in video frames through intelligence-based representation. *Complex & Intelligent Systems*, 2.

Yang, S., Luo, P., Loy, C. C. e Tang, X. (2016). Wider face: A face detection benchmark. Em *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yuan, Y., Dai, F., Song, Y. e Zhao, J. (2021). On fatigue driving detection system based on deep learning. Em *Proceedings of 2020 Chinese Intelligent Systems Conference*, páginas 734–741.

Zhang, K., Zhang, Z., Li, Z. e Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503.

Zheng, Z., Yang, J. e Yang, L. (2005). A robust method for eye features extraction on color image. *Pattern Recognition Letters*, 26(14):2252–2261.