UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística e Computação Científica

MATEUS CHIQUETO DOS SANTOS

Detecção de Laranjas usando Single-Shot Multibox Detector com Arquitetura Mobilenet

Campinas

2024

Mateus Chiqueto dos Santos

Detecção de Laranjas usando Single-Shot Multibox Detector com Arquitetura Mobilenet

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Matemática Aplicada.

Orientador: Marcos Eduardo Ribeiro do Valle Mesquita

Este trabalho corresponde à versão final da Dissertação defendida pelo aluno Mateus Chiqueto dos Santos e orientada pelo Prof. Dr. Marcos Eduardo Ribeiro do Valle Mesquita.

Campinas

2024

Ficha catalográfica
Universidade Estadual de Campinas (UNICAMP)
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

Santos, Mateus Chiqueto dos, 1999-
Sa59d Detecção de laranjas usando single-shot multibox detector com arquitetura mobilenet / Mateus Chiqueto dos Santos. – Campinas, SP : [s.n.], 2024.

Orientador: Marcos Eduardo Ribeiro do Valle Mesquita.
Dissertação (mestrado) – Universidade Estadual de Campinas (UNICAMP), Instituto de Matemática, Estatística e Computação Científica.

1. Aprendizado de máquina. 2. Inteligência computacional. 3. Visão por computador. 4. Redes neurais convolucionais. I. Mesquita, Marcos Eduardo Ribeiro do Valle, 1979-. II. Universidade Estadual de Campinas (UNICAMP). Instituto de Matemática, Estatística e Computação Científica. III. Título.

Informações Complementares

Título em outro idioma: Orange detection using single-shot multibox detector with mobilenet architecture
Palavras-chave em inglês:
Machine learning
Computer vision
Computation intelligence
Convolutional neural networks
Área de concentração: Matemática Aplicada
Titulação: Mestre em Matemática Aplicada
Banca examinadora:
Marcos Eduardo Ribeiro do Valle Mesquita [Orientador]
João Batista Florindo
Kleber Xavier Sampaio de Souza
Data de defesa: 27-06-2024
Programa de Pós-Graduação: Matemática Aplicada

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0009-0001-1981-1925
- Currículo Lattes do autor: http://lattes.cnpq.br/5527312870805584

Dissertação de Mestrado defendida em 27 de junho de 2024 e aprovada pela banca examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). MARCOS EDUARDO RIBEIRO DO VALLE MESQUITA

Prof(a). Dr(a). JOÃO BATISTA FLORINDO

Prof(a). Dr(a). KLEBER XAVIER SAMPAIO DE SOUZA

A Ata da Defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós-Graduação do Instituto de Matemática, Estatística e Computação Científica.

Este trabalho é dedicado à Deus que me trouxe até aqui,
e à minha esposa Débora, que me sempre me apoiou.

Agradecimentos

Primeiramente, agradeço ao Prof. Marcos Valle pela orientação e paciência ao longo deste tempo. Sou grato aos meus pais, que me apoiaram durante todo o mestrado, e também à minha esposa, que sempre está ao meu lado. Sou grato a Deus pelos meus irmãos de Campinas e São Paulo, que nunca me deixaram sozinho. O presente trabalho foi realizado com o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

"All models are wrong,
but some are useful."
(George Box)

Resumo

A detecção de frutos em imagens desempenha um papel essencial em várias aplicações, como agricultura de precisão, classificação de produtos e controle de qualidade. Nesse contexto, ferramentas de aprendizado de máquina, incluindo as redes neurais profundas, apresentam excelente desempenho e representam o estado-da-arte em problemas de classificação e detecção de objetos em imagens. Este estudo explora a aplicação de redes profundas para a tarefa de detecção de frutos em imagens. O foco está na utilização das redes neurais convolucionais (*CNN*, do inglês: *convolutional neural network*) devido à sua capacidade de aprendizado de características complexas, tornando-as adequadas para identificar e localizar frutos em diversas condições de iluminação e fundos variados. O objetivo deste trabalho é descrever a rede SSDlite e avaliar seu desempenho para detecção de laranjas ainda nas laranjeiras. A SSDlite é um detector de objetos da classe SSD (do inglês, *single shot multibox detector*) que tem como base uma rede convolucional leve concebida para aplicações de visão computacional em dispositivos móveis chamada MobileNet. Os experimentos computacionais indicaram que a rede SSDLite teve um bom desempenho na detecção de frutos. Além disso, a SSDlite apresentou desempenho melhor que a rede SSD original baseada na arquitetura VGG; porém, utilizando um número muito menor de parâmetros.

Palavras-chave: redes neurais convolucionais, inteligência computacional, aprendizado de máquina, visão computacional.

Abstract

Accurate fruit detection in images plays an essential role in various applications, such as precision agriculture, product sorting, and quality control. In this context, machine learning tools, including deep neural networks, present excellent performance and represent state-of-the-art problems in classifying and detecting image objects. This study explores the application of deep networks to fruit detection in images. The focus is on using convolutional neural networks (CNN) due to their ability to learn complex features, making them suitable for identifying and locating fruits in different lighting conditions and varied backgrounds. The objective of this work is to describe the SSDlite network and evaluate its performance in detecting oranges on orange trees. The SSDlite is an object detector from the class SSD (single shot multibox detector) based on a lightweight convolutional network designed for computer vision applications on mobile devices called MobileNet. The computational experiments indicated that the SSDLite network performed well in fruit detection. Furthermore, SSDlite presented better performance than the original SSD network based on the VGG architecture; however, it used a much smaller number of parameters.

Keywords: convolutional neural networks, computational intelligence, machine learning, computer vision.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UNICAMP	Universidade Estadual de Campinas
IMECC	Instituto de Matemática, Estatística e Computação Científica
CNN	Redes Neurais Convolucionais
SSD	Single-Shot Multibox Detector
NN	Neural Network
MLP	Multilayer Perceptron
SGD	Stochastic Gradient Descent
IoU	Intersection over Union
R-CNN	Regions With CNN Features
YoLo	You Only Look Once
ReLU	Rectified Linear Unit
FC	Fully Connected
CSP	Convoluções Separáveis por Profundidade
BN	Batch Normalization
RPN	Region Proposals Networks

Sumário

1 INTRODUÇÃO

Aprendizado de máquina (em inglês, *Machine Learning*) é a ciência de programação de computadores para que eles possam aprender com os dados (GÉRON, 2017). Aprendizado de maquina é utilizado em diversas situações, por exemplo: transcrição de voz, como visto em Shukla, Aanand e Nithiya (2023); classificação de imagens, por exemplo, Lorente, Riera e Rana (2021); carros autônomos, apresentado em Shukla, Aanand e Nithiya (2023); entre outros.

Visão Computacional (em inglês, *Computer Vision*) é a ciência de programar computadores para aprenderem por meio de imagens e vídeos (SZELISKI, 2011).

Aprendizado Profundo (em inglês: *Deep Learning*), está presente dentro do aprendizado de máquina, e são métodos compostos por múltiplos níves de representação, obtidos por uma composição de módulos não-lineares. A profundidade é medida pela quantidade de níveis de representação contida no modelo. Desse modo pode-se aproximar funções ainda mais complexas (LECUN; BENGIO; HINTON, 2015).

Detecção de objetos envolve classificar objetos em imagens e segmentação do objeto, ou seja, classificá-lo e informar a sua localização (AMJOUD; AMROUCH, 2023).

Em geral, os modelos de aprendizado profundo são redes neurais. Uma rede neural artificial, em sua forma mais básica, pode ser definida matematicamente como um modelo que consiste em várias unidades interconectadas, chamadas neurônios artificiais (GÉRON, 2017). Nesta dissertação, focaremos nas redes neurais convolucionais. Uma rede neural convolucional é uma arquitetura de rede neural projetada especificamente para o processamento de dados estruturados em grade, como imagens.

Um tipo de arquitetura de redes neurais convolucionais para detecção de objetos são as chamadas "Tiro Único"(em inglês, *Single Shot*). Essa classe de redes neurais permite identificar diferentes objetos em uma mesma imagem com pouco custo computacional. Os principais modelos dessa técnica são: *You only look once* (YoLo), (REDMON et al., 2016) e *Single-Shot Multibox Detector* (SSD) (LIU et al., 2016).

O objetivo deste trabalho é descrever a rede neural convolucional SSDLite, uma versão da rede SSD com arquitetura MobileNet v2 (SANDLER et al., 2018), e testar seu desempenho para identificação e contagem de frutos visíveis na copa de uma árvore. Um dos principais desafios enfrentados pelos produtores é a estimativa da safra anual, essencial para o planejamento no setor. O método descrito neste trabalho pode substituir métodos manuais de contagem de frutos por uma ferramenta mais rápida, barata e eficaz para estimar a safra anual. Para o aprendizado da rede utilizamos o banco de dados obtido pelo

Fundo de Defesa da Citricultura (Fundecitrus), que é uma organização com objetivo de promover o desenvolvimento da citricultura no país, e processado com recortes e anotações pela Embrapa. O conjunto de dados conta com diversas imagens de copas de laranjeiras, assim é possível treinar um modelo de detecção de objetos para detectar laranjas via imagens que podem facilmente ser obtidas por um telefone celular ou drone.

Atualmente, o estado-da-arte para detecção de objetos é a rede chamada Yolo v8, que implementa melhorias sobre a rede Yolo, proposta inicialmente por Redmon e Farhadi (2016). Porém, no período de início de pesquisa deste trabalho, a SSD era considerada o estado-da-arte e por este motivo, foi escolhida para ser o objeto da pesquisa.

1.1 Revisão da Literatura

O trabalho de Sakka e Ivanovici (2024), indica uma crescente em trabalhos relacionando agricultura e redes neurais profundas, e ainda, esses trabalhos indicam que as tarefas mais comuns são: detecção de ervas daninhas, detecção de doenças, classificação do tipo e da qualidade da cultura, previsão de rendimento e gestão da água. Não existe uma arquitetura determinada como a melhor para essas tarefas, mas os estudos indicam que é necessário o teste com um conjunto de arquiteturas de redes neurais convolucionais (CNN, em inglês: *Convolutional Neural Networks*), e também, sendo necessário ajuste fino e transferência de aprendizado.

No caso de identificação de laranjas, vários estudos demonstraram o avanço das técnicas ao longo do tempo.

O trabalho de NETO et al. (2019) utilizou a rede YOLO para identificar frutos, com um banco de dados cedido pela Fundecitrus. Este estudo indicou um alto potencial na utilização de redes neurais convolucionais para a identificação de laranjas ainda nas árvores.

Em CERQUEIRA et al. (2020), o teste foi realizado com a rede R-CNN (GIRSHICK et al., 2014), que na época era considerada uma abordagem mais robusta do que a YOLO. Os resultados mostraram avanços significativos em relação ao trabalho anterior.

No ano de 2021, SOUSA et al. (2021) utilizou a rede SSD (LIU et al., 2016). Esta abordagem obteve resultados ainda superiores aos anteriores, e a variação da SSD utilizada neste trabalho é a base para esta dissertação.

O estudo mais recente, de Ribeiro et al. (2022), realizado em 2022, testou variações da YOLO. O melhor resultado foi alcançado com a YOLO v5 (NOURA et al., 2021), que apresentou uma precisão mais alta e menor consumo de memória em comparação com os métodos anteriores.

Além disso, Moura et al. (2023) fez uma comparação entre todos os trabalhos realizados com o banco de dados da Fundecitrus e concluiu que a YOLO v5 obtinha o melhor resultado.

Por fim, Gremes et al. (2023) realizou testes semelhantes, mas com outra base de dados. Este estudo também atestou que a utilização da YOLO para identificação de laranjas ainda nas árvores é promissora.

1.2 Visão geral

Este trabalho compreende uma estrutura organizacional que abrange sete capítulos distintos. O segundo capítulo destina-se à revisão de conceitos fundamentais relativos às redes neurais, visando a introdução da rede convolucional Mobilenet (v2) e do modelo SSD. Neste contexto, são abordados aspectos essenciais como a definição de neurônios em redes neurais e a análise de suas camadas. Além disso, são discutidos conceitos relacionados às camadas convolucionais, como convoluções separáveis profundas, blocos residuais, blocos de gargalo invertido e normalização em lote.

No terceiro capítulo, realiza-se uma revisão sobre o treinamento de redes neurais, abordando aspectos como os tipos de treinamento, funções de perda, o processo de retro-propagação e os métodos de otimização associados. Esse capítulo também apresenta métricas de desempenho para detecção de objetos.

O quarto capítulo concentra-se na exploração das arquiteturas de redes profundas, proporcionando uma análise aprofundada dos algoritmos e das operações subjacentes aos modelos convolucionais empregados pela SSD, bem como aqueles que serviram de inspiração para o desenvolvimento deste, como a VGG-16 (SIMONYAN; ZISSERMAN, 2015), Resnet (HE et al., 2016) e Mobilenet v2 (SANDLER et al., 2018).

No quinto capítulo, é realizada uma abordagem específica sobre detecção de objetos, com ênfase na rede SSD. O objetivo deste capítulo é a compreensão do funcionamento detalhado dessa rede, proporcionando uma análise da sua arquitetura rápida e eficiente.

Por fim, o último capítulo consiste nas considerações finais do trabalho, onde são discutidas as expectativas alcançadas e os resultados obtidos com a utilização da rede em conjunto com o banco de dados.

2 Conceitos sobre Redes Convolucionais

O desenvolvimento das Redes Neurais Convolucionais decorreu do desenvolvimento das redes neurais tradicionais, que sugiram desde o desenvolvimento dos neurônios artificiais. O neurônio artificial é inspirado no neurônio biológico. O neurônio biológico é composto de um corpo celular que contém o núcleo, dendritos, que são ramificações, e axônio. O axônio se divide em outros ramos, chamados telodendros (ou terminais) e em sua extremidade, as sinapses, que conecta-se com outros neurônios. O neurônio biológico recebe impulsos elétricos de outros neurônios através das sinapses, e dispara seu próprio sinal quando recebe estímulos suficientes. O cérebro consiste em um grande número de neurônios interligados que inspiram modelos computacionais. Os modelos inspirados nas conexões dos neurônios são chamados de Redes Neurais Artificiais (ANN, em inglês: *Artificial Neural Networks*), e surgiram com os trabalhos de Mcculloch e Pitts (1943) e Rosenblatt (1958).

Redes neurais convolucionais são um tipo de redes neurais eficientes para processar sinais e imagens. As redes neurais convolucionais podem processar dados em uma dimensão, como sinais (GAMA et al., 2019), textos (AMIN; NADEEM, 2018), ou em duas dimensões como imagens (KRIZHEVSKY; SUTSKEVER; HINTON, 2017) ou espectrogramas (DORFLER; BAMMER; GRILL, 2017), como visto em LeCun, Bengio e Hinton (2015). As redes convolucionais utiliza "filtros", que extraem características de interesse no processamento de imagens. Essa vantagem permite as redes convolucionais aprenderem características como formas, texturas e localizações na imagem (LECUN et al., 1989).

Neste capítulo, serão apresentados detalhes do modelo do neurônio artificial, bem como uma análise das diferentes camadas que o compõem, destacando suas características e as funções desempenhadas na interação entre elas. Para a implementação de redes neurais convolucionais, serão empregadas técnicas como convoluções separáveis por profundidade, blocos residuais e blocos de gargalo invertido. Portanto, serão abordados nesse contexto os fundamentos e as razões subjacentes à escolha desses tipos de convoluções. O propósito deste capítulo é estabelecer uma compreensão sólida dos conceitos fundamentais das redes convolucionais, que servirão de base para as discussões nos capítulos seguintes.

2.1 Redes Neurais e o Modelo do Neurônio

Redes Neurais Artificiais surgiram com o trabalho de Mcculloch e Pitts (1943). Os dois desenvolveram um modelo matemático simples do neurônio, que agrega os sinais de entrada e dispara um sinal caso o valor exceda um limiar dado. Esses neurônios conectados realizaram operações lógicas como E, OU e a negação. Para descrever a atividade de um

neurônio nesses dois estados utiliza-se a função degrau com comportamento observado na Figura 1. A função degrau é dada por:

$$degrau(x) = \begin{cases} 1, & se \quad x \geq 0, \\ 0, & se \quad x < 0. \end{cases} \tag{2.1}$$

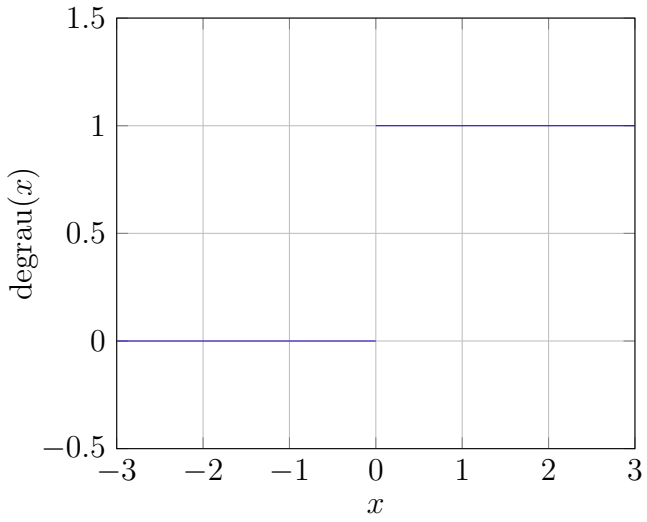


Figura 1 – Gráfico da função Degrau(x)

Em termos matemáticos, sejam as entradas $x_1, x_2, ..., x_n$, os pesos $w_1, w_2, ..., x_n$ e t o limiar pré-definido. A saída y do neurônio de MCulloch e Pitts é dada por:

$$y = \begin{cases} 1, & se \quad \sum_{i=1}^{n} w_i x_i \geq t \\ 0, & \text{caso contrário} \end{cases} \tag{2.2}$$

O psicólogo Frank Rosenblatt desenvolveu, em 1957, o segundo modelo importante de um neurônio artificial, chamado Perceptron (ROSENBLATT, 1958). O Perceptron é semelhante ao modelo desenvolvido por McCulloch e Pitts, porém os pesos são aprendidos por meio de entradas passadas sucessivamente, minimizando a diferença entre a saída desejada e a resposta do neurônio.

Em 1959, Bernard Widrow e Marcian Hoff desenvolveram o modelo chamado *"Multiple Adaptative Linear Elements"* (MADALINE) (WIDROW; LEHR, 1990), a primeira rede neural aplicada com sucesso a um problema real. Uma das principais diferenças entre MADALINE e Perceptron se encontra no aprendizado. Eespecificamente o modelo desenvolvido por Widrow e Hoff aprende por meio do erro de sua saída, dando início ao aprendizado por meio de gradiente.

Outro trabalho relevante foi desenvolvido por Rumelhart, McClelland e Group (1986), no qual foi apresentado o algoritmo da retro-propagação. O algoritmo de retro-propagação é essencial para o treinamento das redes neurais artificiais, permitindo que os pesos das conexões entre neurônios sejam ajustados de forma a minimizar o erro na saída da rede.

Matematicamente, as redes neurais progressivas são constituídas por uma composição de funções, sendo cada uma dessas funções representada por uma camada. As camadas são conjuntos de neurônios que processam e transmitem informações em paralelo. Formalmente, se $\mathcal{N}$ é uma função que representa a rede neural, então ela pode ser expressa como:

$$\mathcal{N}(x) = C^{(n)}...(C^{(2)}(C^{(1)}(x))),$$

onde $C^{(1)}$ é denominada a camada inicial da rede ou de entrada, $C^{(k)}$ é a k-ésima camada da rede para $1 \leq k \leq n$ e $C^{(n)}$ é a camada de saída. A quantidade de camadas determina a profundidade da rede. Cada camada realiza transformações nos dados utilizando seus neurônios. Dessa forma, a saída de uma camada é a entrada para a camada subsequente, resultando em uma sucessão de transformações conforme os dados percorrem a rede.

Uma camada densa, também conhecida por totalmente conectada (FC, em inglês *Fully Connected*), é uma camada onde todas as conexões possíveis podem estar presentes. Na Figura 2 vemos um bom exemplo de camadas densas em uma MLP que é uma composição de camadas densas. As camadas densas desempenham um papel vital em uma rede neural. Porém, o número de pesos gerado é o produto entre quantidade de entradas e a quantidade de neurônios, resultando em alto custo computacional (BASHA et al., 2020).

Seja $\mathbf{x}$ o vetor de entrada da camada densa, $\mathbf{W}$ a matriz de pesos da camada, $\mathbf{b}$ o vetor de bias da camada, e "f" a função de ativação. A saída y de uma camada densa é dada por:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.3}$$

Nos modelos de aprendizado de máquina mais utilizados para regressão ou classificação, as últimas camadas são densas.

O Teorema da Aproximação Universal estabelece uma importante propriedade das redes neurais de camadas densas, com saída linear e pelo menos uma camada densa oculta com funções de ativação ReLU ou sigmoide. Segundo este teorema, essas redes são capazes de aproximar qualquer função contínua definida em um conjunto compacto (HAYKIN, 1998). O Teorema da Aproximação Universal diz que, dada uma função contínua $\mathcal{N} : K \to \mathbb{R}^m$, onde $K \subseteq \mathbb{R}^n$ é um conjunto compacto, e dado $\epsilon > 0$, existe uma MLP $\mathcal{N}^* : \mathbb{R}^n \to \mathbb{R}^m$ tal que

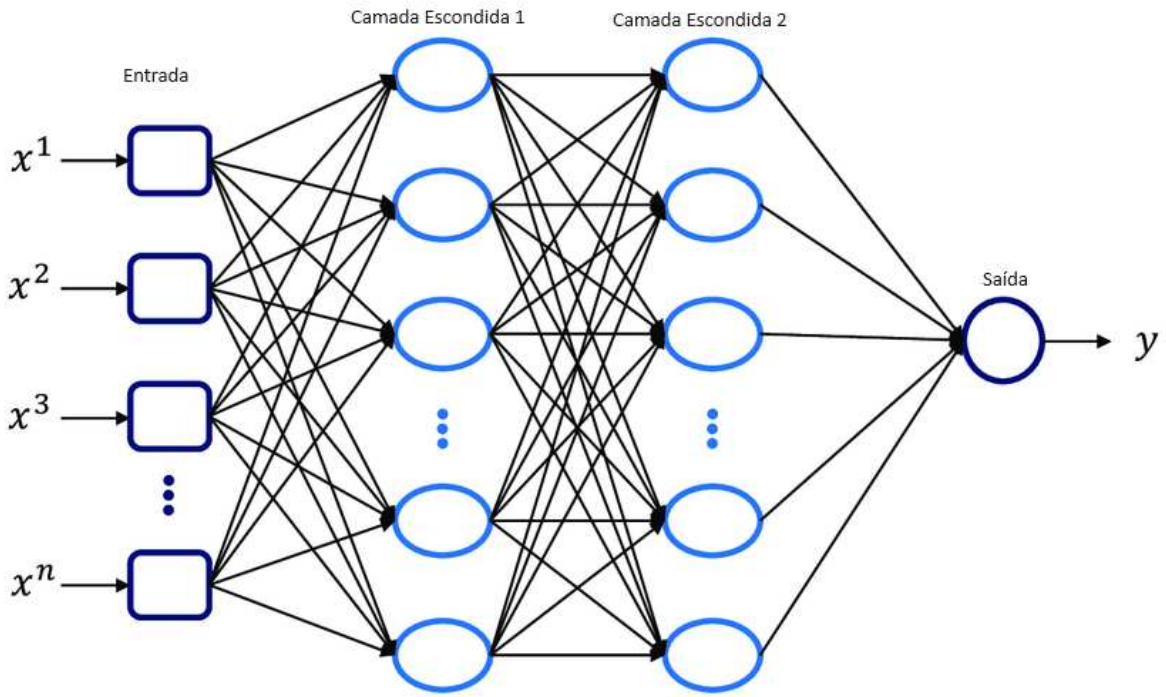


Figura 2 – Arquitetura de uma MLP com duas camadas escondidas. Fonte: Shirazi e Frigaard (2021)

$$\|\mathcal{N}(x) - \mathcal{N}^*(x)\| \leq \epsilon, \quad \forall x \in K.$$

Este resultado fundamenta a capacidade das MLP se aproximarem diversas funções, oferecendo assim uma ferramenta poderosa em campos como reconhecimento de padrões, processamento de linguagem natural e visão computacional, entre outros.

O Perceptron Multicamadas (MLP, em inglês: *Multilayer Perceptron*) é composto de uma ou mais camadas ocultas, uma camada de entrada e uma de saída. O vetor de entrada é processado pelo MLP de modo progressivo (em inglês, *feedfoward*), passando por cada camada. Na Figura 2 é possível ver que um neurônio de uma certa camada é conectado com todos os neurônios da camada posterior. Durante o treinamento da rede neural, os pesos são continuamente adaptados pela minimização do erro de treinamento entre as saídas desejadas e as saídas computadas.

As funções de ativação tem papel fundamental nas ANNs, principalmente nas redes MLP. Existem dois tipos de funções de ativação mais utilizados:

Funções Sigmoide: No início, a maioria das redes neurais utilizavam funções sigmoides, dadas pela função logística e a função tangente hiperbólica (GOODFELLOW;

BENGIO; COURVILLE, 2016). A função logística sigmoide é dada por:

$$\text{logística}(x) = \frac{1}{1 + e^{-x}}, \tag{2.4}$$

e a função tangente hiperbólica, dada por:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{2.5}$$

Essas duas funções são saturadas por entradas com valores muito altos ou muito baixos, como é possível ver na Figura 3. Essa característica das funções sigmoide leva ao problema de desaparecimento do gradiente, que será abordado em outra seção.

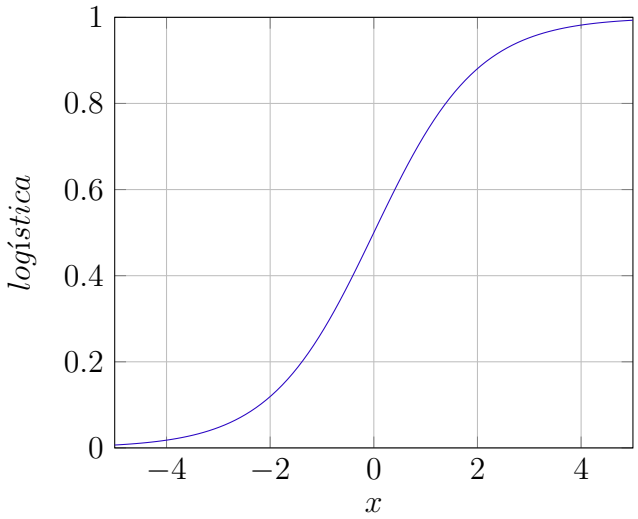


Figura 3 – Gráfico da função Logística(x)

ReLU : Unidade retificada linear ($ReLU$) é uma função linear por partes. Ela retorna 0 para valores negativos e a identidade para valores positivos. O seu gradiente assume somente valores entre 1 e 0 (como mostra o gráfico na Figura 4), o que resolve o problema de desaparecimento do gradiente causado pelas funções sigmoide. A função $ReLU$ é dada por,

$$ReLU(x) = \max(0, x). \tag{2.6}$$

Softmax: Na *Softmax* o vetor é transformado em valores entre 0 e 1, podendo ser interpretados como probabilidades. A função Softmax é utilizada principalmente na camada de saída da rede neural quando aplicada em problemas de classificação multiclasse. Seja V um vetor de valores reais e V_i a sua i-ésima entrada, temos que:

$$[softmax(\mathbf{V})]_i = \frac{\exp V_i}{\sum_{j=1}^{n} \exp V_j}, \forall i = 1, ..., n. \tag{2.7}$$

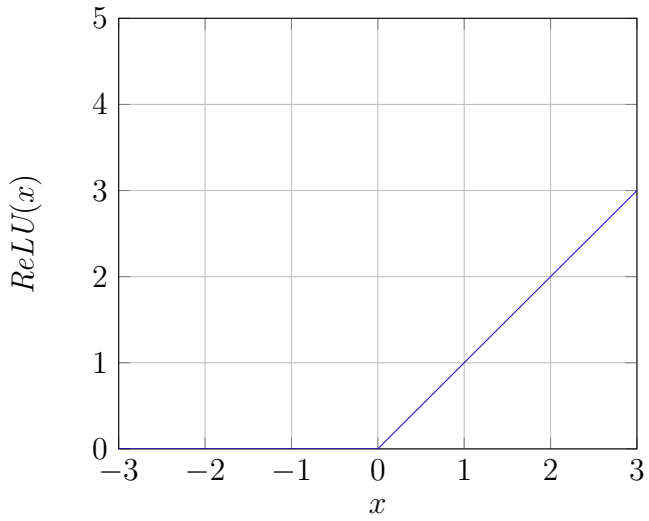


Figura 4 – Gráfico da função $ReLU$

2.2 Camadas Convolucionais

As camadas convolucionais são inspiradas no campo receptivo local do córtex visual e são os blocos mais importantes de uma rede neural convolucional.

Os neurônios do córtex visual contêm um campo receptivo local (HUBEL; WIESEL, 1959). Isto é, os neurônios têm campos visuais pré-determinados para reagir a determinados estímulos. Assim, neurônios de diferentes campos podem se unir e formar todo o campo visual. Também podem existir neurônios de diferentes níveis de características, assim os neurônios de nível superior obtêm informações pelos neurônios de níveis inferiores. Baseado nessas informações, os neurônios de uma camada convolucional se conectam com as entradas de uma certa região na imagem de entrada, ou seja, no campo receptivo local.

Os neurônios no córtex visual inspiram a arquitetura das redes neurais convolucionais, que processam imagens de forma similar, usando filtros para extrair características locais. Para imagens em escala de cinza, a representação é uma matriz bidimensional; para imagens coloridas, um tensor tridimensional que reflete a profundidade dos canais de cor. Essa estrutura permite que a rede combine informações de diferentes canais, formando uma representação completa da imagem.

Para ser mais específico: Um neurônio na linha i, coluna j está conectado às saídas dos neurônios da camada anterior localizado nas linhas i a $i + D_f - 1$, colunas j a $j + D_f - 1$, sendo D_f a dimensão do filtro convolucional, que representa o tamanho do campo receptivo local.

Para efetuar a operação de convolução, é utilizado o *kernel*, denotado por W. O *kernel* é um tensor utilizado para extração de características em uma imagem. A Figura 5 mostra três casos em que são representados no plano superior a camada resultante da

convolução e no plano inferior a camada que sofrerá a convolução. A sombra presente no plano ilustra o *kernel* da convolução.

A ideia principal é utilizar os *kernels* para encontrar características que serão armazenadas nos mapas de características, que serão as novas entradas das próximas camadas. Antes de apresentarmos matematicamente uma convolução são necessários dois conceitos: *Padding e Stride*.

1. *Padding* (em português: preenchimento) é uma técnica utilizada para preservar as dimensões espaciais da imagem de entrada após as convoluções. Envolve adicionar entradas extra na borda do mapa de características de uma entrada antes da convolução. O tipo mais comum de *padding* é o chamado *zero padding*. *Zero Padding* consiste em adicionar "zeros" ao redor das entradas para que uma camada tenha a mesma altura e largura da camada anterior. Essa ferramenta permite que a largura e altura do mapa não sejam dependentes do tamanho do *kernel*. Os filtros convolucionais podem se relacionar com o mapa de características por três maneiras diferentes utilizando *zero padding*, como também podem ser vistos na Figura 5:

   Convolução "*valid*": Não é utilizado padding. A saída será de alguma maneira mais regular que as outras, porém, com dimensão menor;

   Convolução "*Same*": A convolução utiliza o padding quando necessário para manter o tamanho da saída igual da entrada.

   Convolução "*Full*": Supondo que o filtro tenha dimensões $k \times k$, cada entrada da borda é visitada por $'k'$ vezes, uma em cada posição possível no filtro.

2. *Stride* é o nome dado entre a distância entre os campos receptivos visitados pelo filtro. Um neurônio em (i, j) está ligado às saídas dos neurônios na camada anterior nas linhas $(i \times s, j \times s)$ a $(i \times s + D_f - 1, j \times s + D_f - 1)$, tal que D_f é o tamanho do campo receptivo do *kernel* de K e s o tamanho do *stride*. A Figura 6 ilustra uma convolução com *stride* de tamanho $s = 2$.

Agora, vejamos matematicamente uma convolução: Seja X o tensor representando uma imagem de entrada e W o *kernel*. Um neurônio convolucional é dado por (GÉRON, 2017):

$$Z_{i,j,k} = b_k + f\Big(\sum_{u=0}^{D_f-1}\sum_{v=0}^{D_f-1}\sum_{k'=0}^{m-1} X_{i',j',k'} \times W_{u,v,k',k}\Big), \tag{2.8}$$

com: $\begin{cases} i' = i \times s + u \\ j' = j \times s + v \end{cases}$, em que s é o tamanho do *stride* no campo; "f" é a função de ativação; b_k é o viés do mapa k; m é o número de mapas na entrada da camada; e $W_{u,v,k,k'}$ é o componente ou peso do *kernel*.

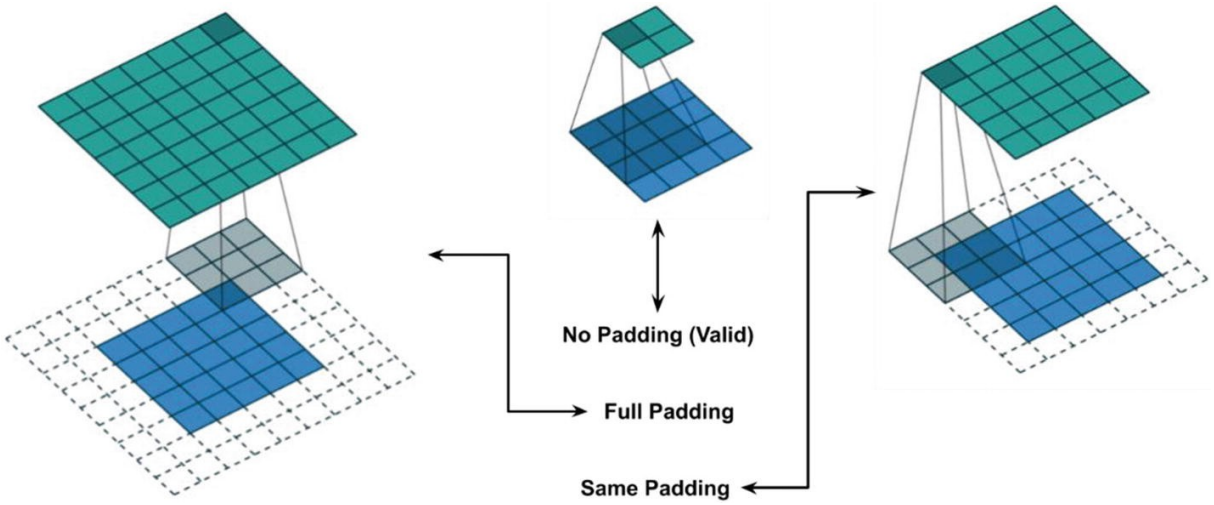


Figura 5 – Os três tipos de padding. Fonte: (EL-AMIR; HAMDY, 2019)

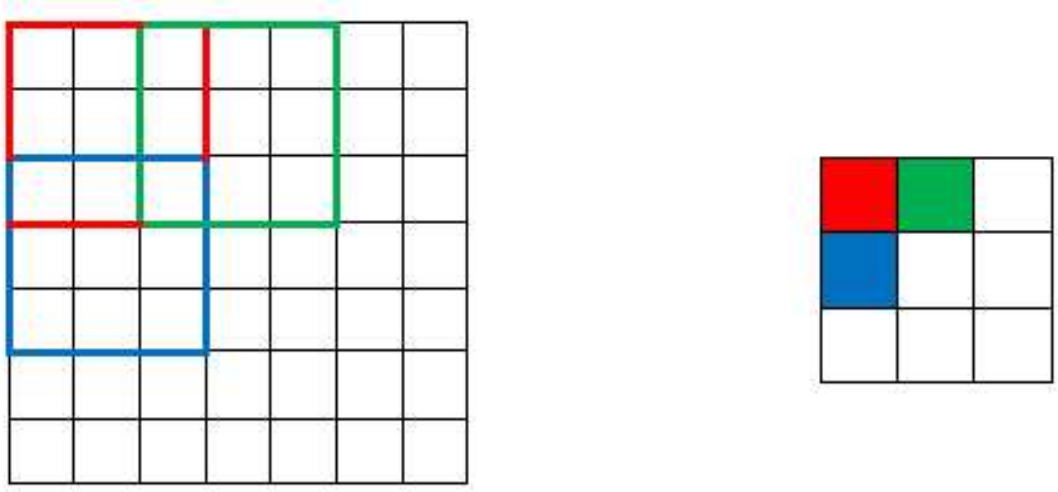


Figura 6 – Um exemplo de *stride*. É possível notar à direita os campos receptivos e à esquerda a camada posterior.

É importante destacar que a quantidade de canais é a quantidade de *kernels* de convolução que serão aplicados. Logo, se aumenta-se o número de *kernels*, aumenta-se o número de canais saída.

2.3 Camadas *pooling*

No *pooling*, ou subamostragem, um mapa de características se conecta com um mapa de grau menor. A camada *pooling* transforma a saída utilizando uma operação estatística entre a vizinhança de uma entrada. Ao longo da rede, geralmente aumenta-se o número de mapas e diminui a resolução das entradas.

O *pooling* pode ser utilizado para a rede obter sensibilidade a alterações nos

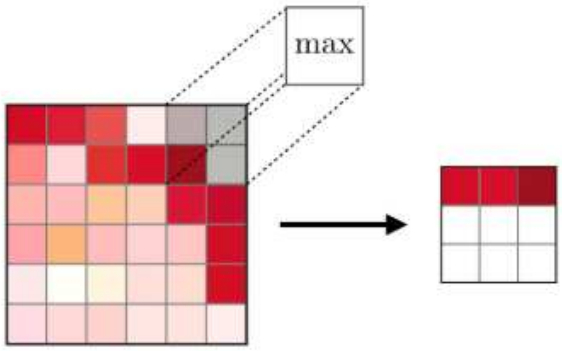


Figura 7 – Um exemplo de max pooling.

objetos e preservar a localização de características (HE et al., 2016).

Uma unidade de *max pooling* calcula o máximo de uma região no mapa de características, isso ajuda ao modelo a ignorar pequenos deslocamentos ou distorções. Um exemplo de max pooling está na Figura 7. Outro tipo de *pooling*, o *average pooling* seleciona a média entre as entradas, isso reduz o efeito de ruídos, mas atribui a mesma importância a todos elementos da região que está sofrendo o *pooling*, isso pode reduzir o poder de discriminação do modelo. Já o *max pooling* seleciona o maior valor, e evita atribuir importância a características desnecessárias, porém pode capturar ruídos. Em geral, o *pooling* pode ajudar a CNN a aprender atributos invariantes e também a reduzir a complexidade computacional reduzindo a resolução das características enquanto preserva as informações mais relevantes (NIRTHIKA et al., 2022).

2.4 Convoluções Separáveis por Profundidade

Um dos principais problemas das CNNs é o número alto de parâmetros e a complexidade computacional. A ideia é substituir convoluções regulares por dois passos mais simples, resultando nas chamadas convoluções separáveis por profundidade. Especificamente, uma Convolução Separável por Profundidade (CSP, em inglês *Depthwise Separable Convolutions*) é uma fatoração de uma convolução tradicional. Em convoluções, a primeira em profundidade (em inglês: *depthwise*), e a seguinte, chamada 'pontual' (em inglês: *pointwise*), como pode ser visto na Figura 8.

A principal motivação para fatorar a convolução é reduzir o número de parâmetros e o número de parâmetros efetuados pela rede neural (Jin; Dundar; Culurciello, 2014). Com efeito, uma CNN muitas vezes pode conter redundância em sua representação pelos pesos. A redundância gera maior custo computacional no treinamento.

Considere uma imagem de dimensão $(D_k \times D_k \times M)$. Em uma convolução padrão, para cada canal, é aplicado um filtro $D_f \times D_f$ e somam-se os mapas de características,

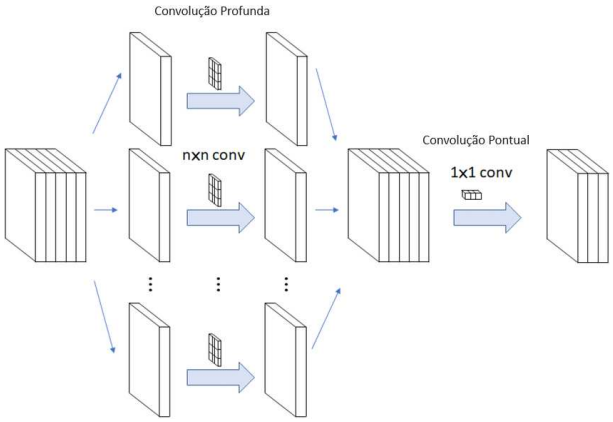


Figura 8 – Estrutura de uma CSP. Fonte: Junejo e Ahmed (2021).

resultando em $D_k \times D_k \times M \times D_f \times D_f$ operações. Seja N o número de mapas de características de saída. Logo, serão efetuadas $D_k \times D_k \times M \times D_f \times D_f \times N$ operações.

Já em uma convolução fatorada, a convolução em profundidade irá aplicar um filtro em cada canal, efetuando $D_k \times D_k \times M \times D_f \times D_f$ operações. A primeira camada somente filtra, não combina as saídas para criar novas características. Após isso, a convolução pontual, irá, separadamente, efetuar uma combinação linear dos resultados encontrados das convoluções em profundidade. Desse modo, sendo N o número de entradas, são efetuadas $M \times N \times D_f \times D_f$ operações. Assim, uma convolução separada em profundidade realiza ao todo: $D_k \times D_k \times M \times D_f \times D_f + M \times N \times D_f \times D_f$ operações. Convoluções separadas utilizam cerca de 8 a 9 vezes menos custo operacional que convoluções padrão (HOWARD et al., 2017). Isso também indica que a CSP gera menos parâmetros que a convolução tradicional, e como consequência necessita de menos alocação de memória para a inferência da rede, o que aumenta a velocidade da inferência por ter menos operações e permite o processo em dispositivos de recursos limitados. Com efeito, na *Mobilenet*, que utiliza de CSP, são gerados $4.2M$ de parâmetros, já na VGG, que utiliza convolução tradicional, são gerados $138M$ parâmetros (Kaiser; Gomez; Chollet, 2017).

A saída $\hat{Z}_{i,j,k}$ da convolução em profundidade com o filtro $\hat{W}_{u,v,k'}$ para o canal k' é dada por:

$$\hat{Z}_{\hat{i},\hat{j},\hat{k}} = \sum_{u=0}^{D_f-1} \sum_{v=0}^{D_f-1} X_{i',j',k'} \times \hat{W}_{u,v,k'}, \forall k' \in 0, 1, ..., m. \tag{2.9}$$

Em que $\hat{i} = s \times i + u$ e $\hat{j} = s \times j + v$. Seja $W_{k',k}$ o filtro convolucional entre o canal k' e k. A saída $Z_{i,j,k}$ da convolução pontual é dada por:

$$Z_{i,j,k} = \sum_{k'}^{m} \hat{Z}_{\hat{i},\hat{j},\hat{k}} \times W_{k',k}, K \forall 0, 1, .., m-1 \tag{2.10}$$

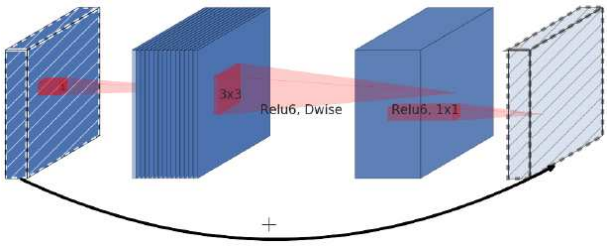


Figura 9 – Estrutura de bloco de gargalo. Fonte: Sandler et al. (2018).

2.5 Bloco Residual

Uma conexão residual define-se por adicionar a identidade à saída de um neurônio, caracterizando os chamados blocos residuais. Os blocos residuais podem contribuir para a resolução do problema do desaparecimento de gradientes (em inglês *Vanishing Gradients*). Desaparecimento de gradientes é um problema que ocorre no treinamento de redes neurais profundas. O problema do desaparecimento de gradientes ocorre quando os pesos de camadas inferiores não são alterados, assim, o treinamento não converge para uma boa solução. No trabalho de Glorot e Bengio (2010), nota-se que um dos causadores desse problema seriam as funções de ativação sigmoide e a distribuição aleatória de pesos, por isso, uma alternativa é utilizar blocos residuais. A função de ativação *ReLU* e uma boa inicialização dos pesos sinápticos podem evitar o desaparecimento de gradientes. O uso de blocos residuais também contribui de forma positiva para contornar esse problema (HE et al., 2016).

Um bloco residual é obtido substituindo um bloco C de uma ou mais camadas por $R(x) = x + C(x)$. Diferentemente dos blocos tradicionais, uma camada alimenta camadas adiante devido ao "*shortcut*" expresso pela adição do termo "x" ao bloco.

2.5.1 Bloco de Gargalo Invertido

O bloco residual pode ser modificado para o formato de gargalo invertido. Para cada função residual R, são utilizadas três camadas. São as convoluções 1×1, 3×3 e 1×1, nessa ordem. A primeira e a última camada são responsáveis por reduzir e restaurar dimensões e contém um número t vezes maior de mapas de características comparado a camada intermediária. Dessa maneira, a camada intermediária é chamada de camada de gargalo, com um número N de canais (HE et al., 2016).

As convoluções separadas profundas podem ser utilizadas com um bloco de gargalo invertido que modifica a ordem das convoluções, expandindo logo na primeira convolução ao invés de comprimir a quantidade de canais (SANDLER et al., 2018). Assim, a primeira é chamada "Camada de Expansão" (camada da esquerda na Figura 9). Essa camada contém uma convolução pontual 1×1, que expande a dimensão da entrada dos

mapas de caraterísticas, e aplica-se uma ativação não linear $ReLU$. Essa expansão segue o hiperparâmetro t, que leva os mapas para tM canais. A segunda camada é a convolução em profundidade 3×3 (camada central na Figura 9) vista na subseção anterior, que filtra espacialmente utilizando um tensor de dimensão maior. Finalmente, volta-se ao número inicial de canais com a terceira convolução, chamada de "Camada de Projeção" ou "Camada de Gargalo" (camada da direita na Figura 9), pois diminui o número dos mapas de características, a camada de gargalo tem tensor 1×1, e utiliza ativação linear.

Em todo esse processo, utiliza-se de conexões *shortcut*, transformando o bloco de gargalo invertido em bloco residual.

A equação de expansão é semelhante à Equação 2.10, porém, aumenta o número de canais. A convolução em profundidade é semelhante à Equação 2.9. E a convolução de gargalo diminui o número de canais, porém, é semelhante à convolução chamada pontual.

Como nas convoluções separáveis em profundidade, o bloco de gargalo invertido diminui a alocação de memória necessária na rede pois mantém a convolução mais eficiente, mantendo-a em número reduzido de canais no estágio inicial e final da convolução, e também, acelera o treinamento com a conexão residual.

2.6 Normalização em lote

A normalização em lote, proposta por Ioffe e Szegedy (2015), pode acelerar o treinamento e melhorar a capacidade de generalização da rede. A técnica consiste em adicionar uma operação no modelo imediatamente antes da função de ativação de cada camada, ou simplesmente centralizar em zero e normalizar as entradas, e então escalonar e deslocar o resultado utilizando os parâmetros "dimensão" e "deslocamento", estimando a média e o desvio padrão das entradas para centralizar e normalizar. Deste modo, o modelo aprende a escala ideal e a média das entradas para cada camada.

Seja o minilote B, n_B o número de amostras em B, X^i as entradas da camada, ε um número pequeno que impede a divisão por zero, chamado de termo de suavização, φ o parâmetro de escala para a camada, β, o parâmetro de deslocamento,

$$\mu_B = \frac{1}{n_B} \sum_{i=1}^{n_B} X^i,$$

a média no minilote B,

$$\sigma_B^2 = \frac{1}{n_B} \sum_{i=1}^{n_B} (X^i - \mu_B),$$

o desvio padrão no minilote B,

$$X_*^i = \frac{X^i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}},$$

a entrada normalizada, e finalmente,

$$Z^i = \varphi X_*^i + \beta,$$

a saída da camada normalizada. Assim, a saída da normalização em lote é uma versão escalonada e deslocada das entradas. Por conta dos parâmetros γ e β, o modulo pode aprender os melhores fatores de forma otimizada, e assim dimensionar os valores atualizados.

A normalização em lote melhora consideravelmente o desempenho de redes neurais profundas, e também em quanto ao problema do desaparecimento dos gradientes, a ponto de poder utilizar funções de ativação que não são a ReLU. Redes sensíveis a alteração podem utilizar taxas de aprendizado maiores, acelerando o processo de aprendizado, porém adicionando complexidade ao modelo.

3 Treinamento e Métricas de Desempenho

Nesse trabalho, o treinamento de uma rede convolucional é baseado em aprendizado supervisionado. No aprendizado supervisionado, os dados de treinamento incluem as soluções desejadas. Para casos de classificação, o modelo deve aprender a classificar novos dados, a partir dos dados rotulados fornecidos a priori. Para casos de regressão, o modelo aprende a prever um alvo determinado a partir de um conjunto de características (GÉRON, 2017).

Durante o treinamento é utilizado uma função para medir a diferença entre os valores da saída da rede e a saída desejada, chamada de função custo. Deste modo, o modelo ajusta seus parâmetros para reduzir o custo, tem-se assim um problema de otimização irrestrita e geralmente são utilizados o gradiente descendente estocástico com retro-propagação.

3.1 Funções Custo

Uma função custo é uma função que mede a diferença entre o que foi predito pela rede e a saída desejada. O objetivo do treinamento é minimizar a função custo. Dois exemplos de função custo são o Erro Quadrático Médio (MSE, em inglês: *Mean Square Error*) e Entropia Cruzada Binária (BCE, em inglês: *Binary Cross Entropy*).

O Erro Quadrático Médio é utilizado em tarefas de regressão e é uma das funções custo mais simples. Sejam n o número de saídas de treinamento, $\bar{y}_i$ a i-ésimo saída do modelo, y_i o i valor alvo do correspondente do conjunto de treinamento. O MSE é dado por:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\bar{y}_i - y_i)^2 \tag{3.1}$$

A entropia cruzada binária (BCE) (SHAN; FANG, 2020) é uma função custo utilizada para classificação binária. O valor de saída da BCE diminui conforme um elemento pertence a classe predita e aumenta quando não pertence. A entropia cruzada binária calcula o custo computando a seguinte média:

$$BCE = -\frac{1}{n}\sum_{n=1}^{n} y_i \log \bar{y}_i + (1 - y_i)\log(1 - \bar{y}_i). \tag{3.2}$$

Para auxiliar na visualização e análise dos resultados, foi utilizado um método de suavização para a curva da função custo do treinamento de um modelo. Especificamente, empregamos a média móvel exponencial (MME), que é uma técnica usada para suavizar

séries temporais, atribuindo mais peso aos valores mais recentes. A fórmula para calcular a MME no período t é dada por:

$$\text{MME}_t = \alpha \cdot x_t + (1 - \alpha) \cdot \text{MME}_{t-1}$$

onde α é o fator de suavização, calculado como:

$$\alpha = \frac{2}{N + 1}$$

em que N é o número de períodos. O valor inicial MME_0 pode ser a média dos primeiros N períodos ou o primeiro valor da série. A MME é eficaz para detectar tendências recentes em dados temporais (MAK, 2021).

3.2 Algoritmos de Otimização

Algoritmos de otimização são métodos matemáticos utilizados para otimização de um modelo. No contexto de redes neurais convolucionais, são utilizados métodos para efetuar a minimização da função custo alterando os pesos do modelo. Para isso é utilizado o Gradiente Descendente que efetua a alteração dos pesos auxiliado pela retro-propagação que calcula o gradiente da função custo.

Gradiente descendente (BOTTOU; CURTIS; NOCEDAL, 2018) é um algoritmo de otimização que ajusta os pesos para minimizar os valores obtidos pela função custo. Seja $Custo(w)$ a função custo a ser minimizada e w os pesos dos neurônios. O gradiente da função custo é um vetor com todas as derivadas parciais, denotado por $\nabla Custo(w)$. Os pontos críticos da função custo serão os pontos onde todas derivadas parciais são iguais a zero.

O gradiente descendente propõe que cada novo peso seja calculado através da equação:

$$w' = w - \epsilon \nabla Custo(w),$$

em que ϵ é a taxa de aprendizado. O método ajusta iterativamente os parâmetros na direção oposta ao gradiente, pois é a direção em que a função decresce mais rapidamente.

O gradiente descendente estocástico (SGD) segue o mesmo processo, porém escolhe um subconjunto aleatório no treinamento em cada etapa e calcula os gradientes por esse subconjunto, sendo vantajoso para conjuntos grandes (GOODFELLOW; BENGIO; COURVILLE, 2016).

Os algoritmos de otimização como SGD e outros como Adam (KINGMA; BA, 2015) são utilizados para atualizar os parâmetros e assim minimizar a função custo. Porém é necessário um algoritmo que calcule o gradiente nos parâmetros na rede. Esse algoritmo é chamado de **retro-propagação** (em inglês: *Backpropagation*) (RUMELHART; HINTON;

WILLIAMS, 1986), que associa a informação da função custo para propagar o gradiente desde a saída à entrada do modelo. O algoritmo consiste em retro-propagar o valor da função custo para os parâmetros da rede. A combinação do gradiente descendente e a retro-propagação é o que efetua o aprendizado do modelo, pois a retro-progação retorna em todos os pesos a derivada da função custo e o gradiente descendente atualiza os pesos, assim efetuando o aprendizado da rede.

3.3 Regularização

Regularização consiste em restringir os pesos de um modelo, para diminuir o sobre-ajuste do treinamento e aumentar a capacidade de generalização. Uma maneira de realizar a regularização é a *Regularização de Tikhonov*ou l_2 (NG, 2004). Essa técnica consiste em adicionar à função custo o termo $\alpha \sum_{i=1}^{n} w_i^2$, tal que w são os pesos do modelo, α é o hiper-parâmetro da regularização e n é a quantidade de pesos. Esse método restringe os pesos pois "força" o modelo a diminuir os seus valores.

"Dropout" (SRIVASTAVA et al., 2014), é outra técnica de regularização. A cada passo do treinamento, um neurônio do modelo tem uma probabilidade de ser inibido. No próximo passo cada neurônio terá a mesma probabilidade de ser inibido. Na Figura 10 é possível observar a estrutura de uma rede neural sem o *dropout* à esquerda e uma rede com esse método à direita. Pelo exemplo é possível notar que existem menos neurônios na rede com *dropout*. A probabilidade de um neurônio ser inibido é chamada de "valor de *dropout*".

O *dropout* aumenta em média 1% a acurácia de um modelo de redes neurais (GÉRON, 2017). Apesar de trazer perda de informações para o treinamento, esse método "força" a rede a treinar os neurônios de forma a não dependerem de outros neurônios. O *dropout* somente é aplicado para a fase de treinamento.

3.4 Métricas de classificação binária

Classificação de objetos é a tarefa onde um modelo identifica a classe de um objeto. As métricas de classificação são um conjunto de medidas utilizadas para checar a efetividade do modelo treinado. Em geral, as detecções são classificadas em: Verdadeiro Positivo (VP), uma detecção classificada corretamente; Falso Positivo (FP), uma detecção que classificou erroneamente; Verdadeiro Negativo (VN), uma detecção que não classificou corretamente como positivo; Falso Negativo (FN), um elemento que deveria ter sido classificado como positivo.

As seguintes métricas são exemplos de medidas de desempenho em problemas de classificação binária:

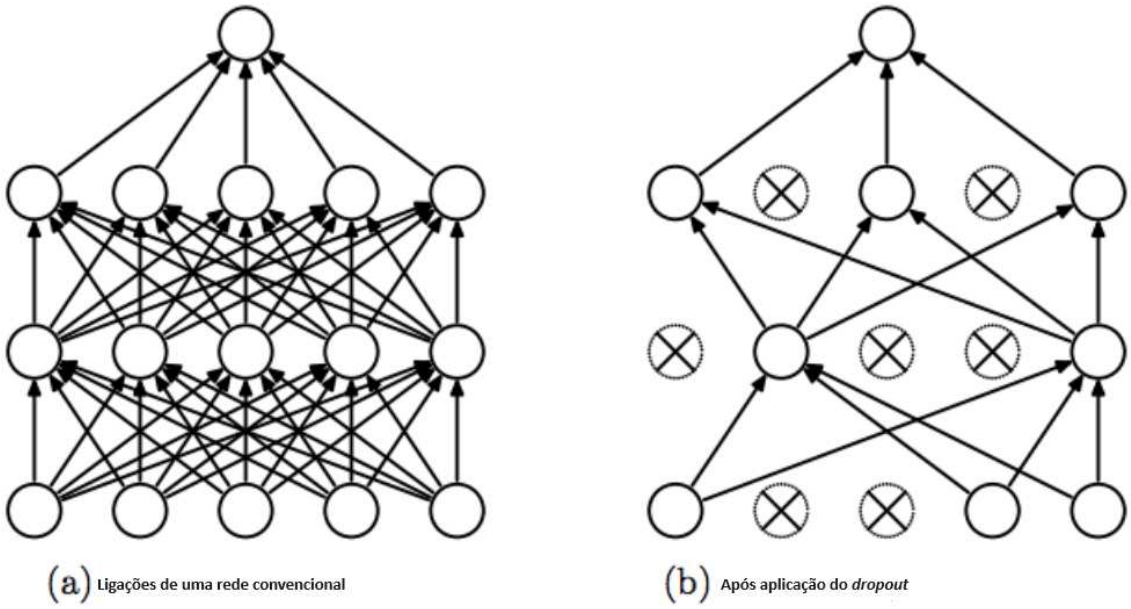


Figura 10 – Exemplo de uma rede neural clássica com dropout. Fonte: (SRIVASTAVA et al., 2014).

1. Acurácia (A_c): A relação de rótulos corretos efetuados pelo modelo, incluindo negativos e positivos.
$$A_c = \frac{VP + VN}{VP + FP + VN + FN}. \tag{3.3}$$
2. Precisão (P_r): A porcentagem de detecções que foram classificadas corretamente.
$$P_r = \frac{VP}{VP + FP}. \tag{3.4}$$
3. Revocação (R_c): A quantidade de objetos positivos que foram classificados corretamente.
$$R_c = \frac{VP}{VP + FN}. \tag{3.5}$$

3.5 Métricas para detecção de objetos

A deteção de objetos tem duas tarefas: Encontrar objetos em uma imagem e identificar a sua classe, por conta disso utiliza-se métricas que contam com outros fatores além do desempenho da classificação binária. Especificamente são utilizados IOU e as métricas dadas pela competição COCO (LIN et al., 2014), Precisão Média (em inglês *Average Precision*, AP) e Revocação Média (em inglês *Average Recall*, AR) e suas variações. A saída de um detector de objetos é uma caixa delimitadora, classe do objeto e número de confiança do objeto para a classe. Por isso, somente precisão, revocação e acurácia não são suficientes para medir a qualidade de um detector.

O índice de *Jaccard* (em inglês: *Jaccard overlap*) ou Intercessão sobre União (IOU, em inglês: *Intersection over Union*) é um índice usado para mensurar a sobreposição

de duas regiões utilizando a área. No caso de detecção de objetos é a relação entre a área da caixa delimitadora de um objeto padrão de um modelo (Gt), e a área da caixa delimitadora de um objeto detectado (Bb) em relação ao Gt, deste modo:

$$IOU = \frac{A(Gt \cap Bb)}{A(Gt \cup Bb)}, \tag{3.6}$$

em que $A()$ denota a área do argumento.

Para as outras métricas são necessárias algumas observações. Podemos definir P_r e R_c em função de um valor confiança (τ) adotado para indicar detecção positiva ou negativa. Dessa maneira consideramos como $VP(\tau)$ apenas detecções positivas corretas com número de confiança maior ou igual a τ, igualmente para $FP(\tau)$. Para $FN(\tau)$ considere os falsos negativos com valor de confiança menor que τ. Desse modo, temos que P_r e R_c em função de τ são dados por:

$$P_r(\tau) = \frac{VP(\tau)}{VP(\tau) + FP(\tau)}; \tag{3.7}$$

e

$$R_c(\tau) = \frac{VP(\tau)}{VP(\tau) + FN(\tau)}. \tag{3.8}$$

Seja T o conjunto de valores de confiança, tais que, $\tau \in T$. Suponha que T é um conjunto de valores ordenados e $\tau_i > \tau_j$, para $i > j$. Dessa forma, $VP(\tau)$ e $FP(\tau)$ são decrescentes para o conjunto T, pois quanto maior o limiar de confiança menores são os valores detectados positivos. Já $FN(\tau)$ é crescente, pois quanto menor o valor de τ, maior o valor de falsos negativos. Dessa forma, pode-se mostrar que R_c é decrescente em τ. Porém, nada pode ser dito sobre a monotonicidade de $P_r(\tau)$. Deste modo, a curva ${P_r}^{\tau} \times {R_c}^{\tau}$, pode ter comportamento de zig-zag em τ pertence a T.

3.5.1 Precisão Média

Precisão Média (AP) não é a média da precisão entre as classes, mas sim a área abaixo da curva $P_r \times R_c$. Quanto mais altos os valores de precisão e revocação, maior será o valor de AP. Para calcular a AP é necessário aproximar a curva $P_r \times R_c$ por uma curva monótona.

A aproximação pode ser dada pela função contínua $\dot{P}_r(R)$, onde $R \in [0, 1]$. A precisão interpolada é dada por (PADILLA et al., 2021):

$$\dot{P}_r(R) = \max_{\tau | R_c(\tau) \geq R} P_r(\tau). \tag{3.9}$$

O valor da precisão interpolada corresponde à máxima precisão em que o valor de R_c é maior que R.

Agora, defina um conjunto ordenado de valores de revocação, denotado por (Υ), tal que $R_c(\tau) \in \Upsilon$. Sabemos que os valores de revocação são decrescentes em relação a τ. Logo, $R_c(\tau_i) < R_c(\tau_j)$, para $\tau_i > \tau_j$. Dessa forma, é possível calcular o valor de AP utilizando os valores de revocação ordenados do conjunto Υ, tal que, $\#\Upsilon = \upsilon$.

$$AP = \sum_{i=0}^{\upsilon} (R_c(i) - R_c(i+1))\dot{P}_r(R_c(i)). \tag{3.10}$$

Outra aproximação para o AP é a aproximação com interpolação de N pontos, onde o conjunto de valores de revocação Υ tem seus valores divididos em pontos igualmente espaçados em um intervalo $[0, 1]$. Assim, cada valor de revocação $R_c(i)$, $i = 1, 2, ..., N$, será dado por:

$$R_c(i) = \frac{N - i}{N - 1}. \tag{3.11}$$

Logo, a expressão de AP será dada por:

$$AP = \frac{1}{N} \sum_{i=1}^{N} \dot{P}(R_c(i)). \tag{3.12}$$

Com essas duas expressões para a precisão média é possível obter algumas variações, como vistas a seguir.

3.5.1.1 Variações da Precisão Média

1. AP@.5 e AP@.75: Nesse caso, a interpolação ocorre para $N = 101$ pontos de revocação, utilizando a Equação 3.12. A diferença entre AP@.5 e AP@.75, está no limite de IoU; o primeiro utiliza $IoU = 0.5$ e o segundo $IoU = 0.75$. Essas duas métricas são comumente utilizadas pelo banco de dados COCO.

2. AP@[.05 : .5 : .95]: É uma expansão do método anterior, pois calcula a AP para 10 valores de IOU e finaliza com a média de AP entre esses valores. A AP@[.05 : .5 : .95] é afetada por diferentes valores de IOU, logo não é possível observar o comportamento de apenas um valor de IOU por essa métrica.

3. AP_s, AP_m, AP_l: São AP sobre escalas, pois aplica a métrica anterior tomando em consideração o tamanho dos objetos.

   a) AP_s: area $< 32^2$ pixels
   b) AP_m: 32^2 pixels < area $< 96^2$ pixels
   c) AP_l: area $> 96^2$ pixels

A interpolação remove o comportamento não monotônico em $P_r \times R_c$ antes de calcular a área sob a curva. Logo, a interpolação por 101 pontos leva à aproximação mais fiel para a curva do que com 11 pontos. Por isso, em geral, os bancos de dados COCO são avaliados com essas métricas. Neste trabalho são utilizados as métricas AP@.5 e AP@.75, sendo a primeira a principal.

3.5.1.2 Exemplo Numérico

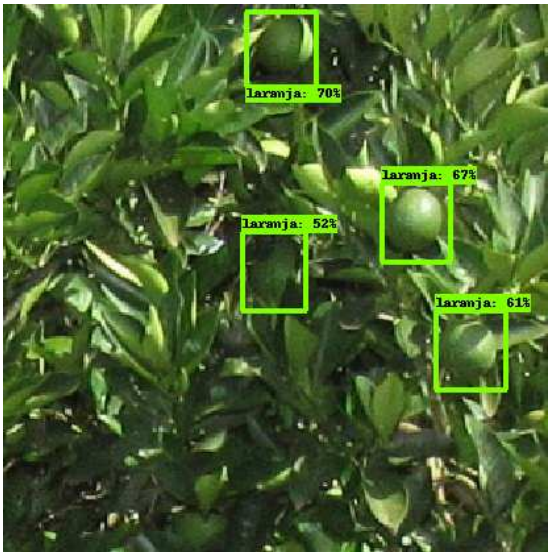


Figura 11 – Detecções

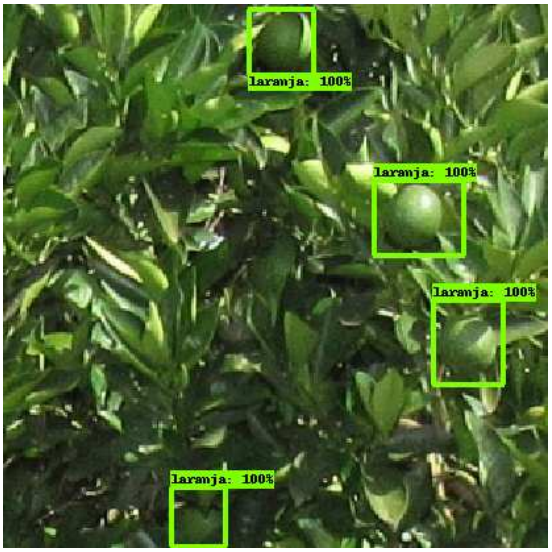


Figura 12 – Caixas Padrão

Figura 13 – Detecções

Figura 14 – Caixas Padrão

Iremos calcular a métrica AP@0.5 nos exemplos dados na Figura 11, Figura 13 e Figura 15. Pelas imagens, vemos à esquerda as detecções e seus valores de confiança, à direita vemos as caixas padrão, nas Figura 12, Figura 14 e Figura 16, que são os alvos de detecção da rede. Em todas as imagens os objetos são da mesma classe:"laranjas" O limiar de IOU escolhido foi de 0.5, que foi utilizado para estabelecer os objetos delimitados por retângulo, que foram detectadas como VP ou FP, verdadeiros positivos e falsos positivos, e as laranjas que não foram detectados ou são contabilizadas como FN.

Pela Equação 3.9 e Equação 3.12, o AP é uma métrica que relaciona precisão e revocação em diferentes valores de confiança. Para isso, é necessário calcular os valores

Figura 15 – Detecções

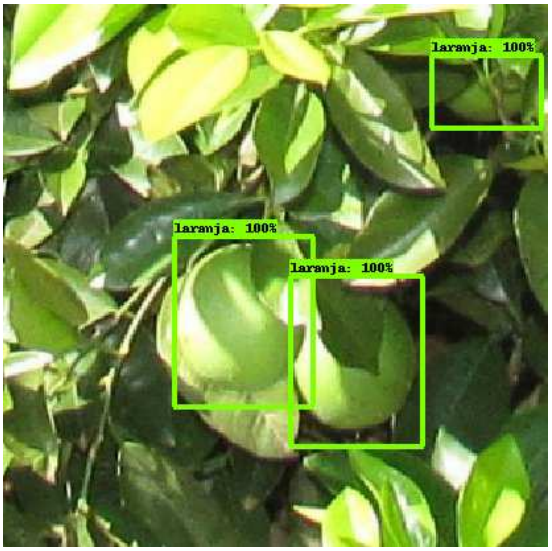


Figura 16 – Caixas Padrão

Confiança (τ)	$VP(\tau)$	$FP(\tau)$	$P_r(\tau)$	$R_c(\tau)$
0.85	1	0	1	0.091
0.77	2	0	1	0.182
0.75	3	0	1	0.272
0.70	4	0	1	0.363
0.70	5	0	1	0.454
0.69	6	0	1	0.545
0.67	7	0	1	0.636
0.61	8	0	1	0.727
0.52	8	1	0.8809	0.727

Tabela 1 – Valores de precisão e revocação por valores de confiança.

de $VP(\tau)$ e $FP(\tau)$ para diferentes valores de confiança nas detecções. Na Tabela 1 temos as detecções e assim temos os valores de precisão por revocação, onde as duas últimas colunas são calculadas conforme Equação 3.7 e Equação 3.8, respectivamente.

Na Figura 17 a seguir podemos observar a curva $Precisão \times Revocação$ para cada valor de confiança obtido.

Finalmente, é possível calcular a interpolação de acordo com a Equação 3.9. Iremos calcular para $N = 11$, assim, dividiremos o eixo x em 11 valores, formando o conjunto $0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$. Perceba que a única mudança ocorrerá para $R_c(0.52) = 0.727$, assim $\dot{P}_r(0.7) = \max_{\tau|R_c(\tau)\geq R} P_r(\tau) = 1$. Na Figura 18 vemos o gráfico da $Precisão \times Revocação$ após a interpolação é:

Finalmente, é possível calcular o valor de AP para $IOU = 0.5$ e $N = 11$. Com

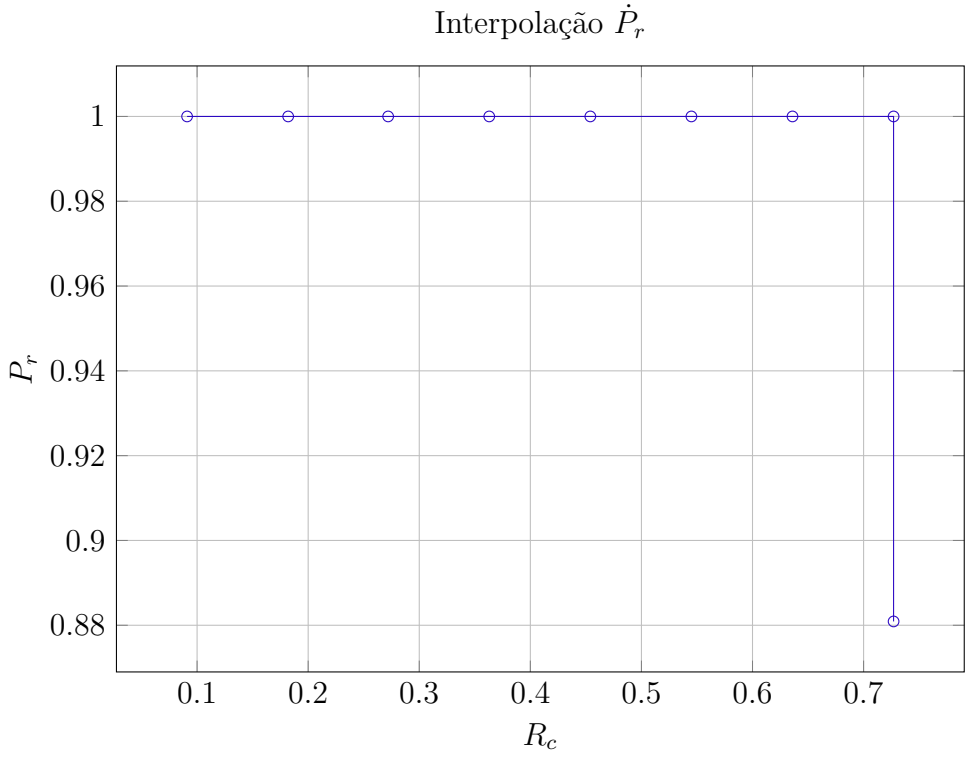


Figura 17 – Gráfico de $Precisão \times Revocação$

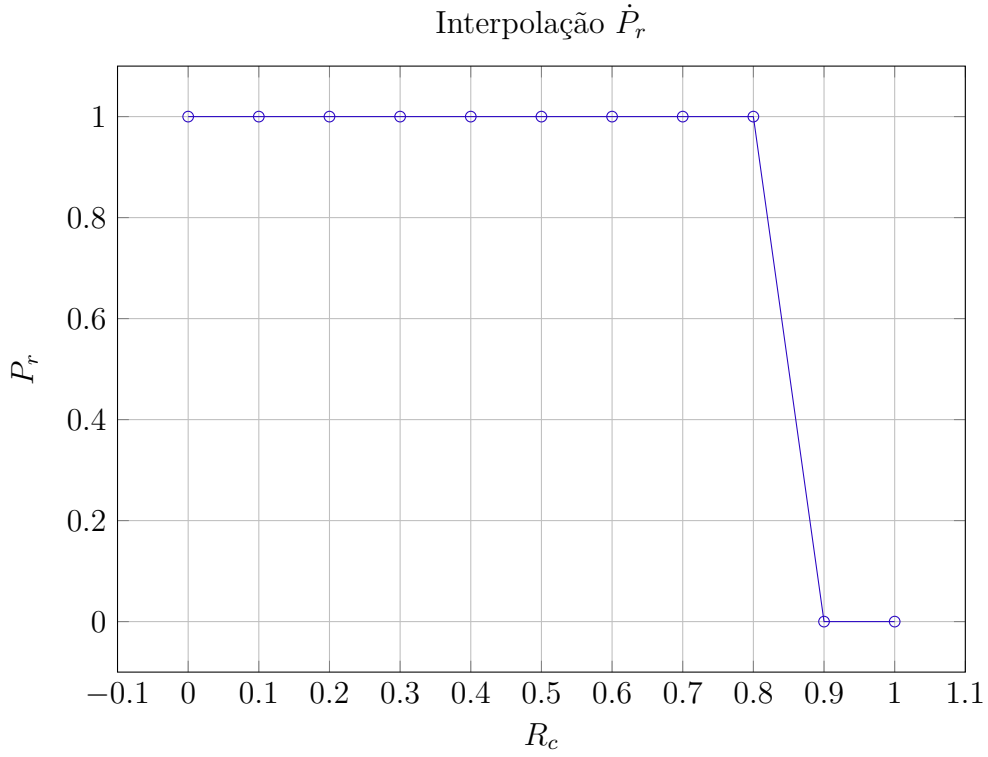


Figura 18 – Gráfico de Interpolação $\dot{P}_r$

efeito, pela Equação 3.12,temos:

$$\begin{aligned} AP = \frac{1}{11}\sum_{i=1}^{11}\dot{P}(R_c(i)) &= \frac{1}{11} * (1+1+1+1+1+1+1+1+1+1+0+0) \\ &= \frac{1}{11} * 9 \\ &= 0.818. \end{aligned} \tag{3.13}$$

Portanto, nas imagens selecionadas, o valor de AP foi 0.818.

3.5.2 Revocação Média

A revocação média (em inglês *Average Recall*, AR) é utilizada para medir a assertividade de um detector de objetos para uma classe. A AR pode ser calculada como a média dos valores máximos de revocação entre os valores de IOU.

Defina um conjunto de O valores de IOU, de modo que $t(o)$ é um valor de limite de IOU. Seja, $P_{t(o)}(\tau), R_{t(o)}(\tau)$ os pontos de precisão e revocação para $IOU = t(o)$. Uma das maneiras de estimar AR através da equação (PADILLA et al., 2021):

$$AR = \frac{1}{O}\sum_{o\in O}\max_{\tau|P_{t(o)}(\tau)>0} Rc_{t(o)}(\tau). \tag{3.14}$$

Assim, a AR é a média dos maiores valores de revocação que a precisão é maior do que 0 para cada valor de IOU.

Para a revocação média, existem variações: AR_1, AR_{10} e AR_{100}. São aplicações de AR que limitam o número de detecções por imagem, ou seja, elas calculam a AR tendo um número fixo máximo de detecções por imagem, seu índice indica a quantidade de detecções por imagem.

3.6 Aumento de Dados

Aumento de dados é uma das melhores maneiras para incrementar a capacidade de generalização de um modelo de aprendizado de máquina. Aumento de dados tem sido bastante efetivo para detecção de objetos pois simula grande quantidade de variações nas imagens.

Operações como translação, rotação, mudança de dimensão podem ajudar o modelo à aprender características das variações nas imagens e assim melhorar a generalização do modelo (GOODFELLOW; BENGIO; COURVILLE, 2016). Na Figura 20 vemos as imagens geradas por aumento de dados utilizando a imagem mostrada na Figura 19. No exemplo, o “Aumento 1” chamado de“Espelhamento Horizontal", é obtido espelhando a imagem original com respeito ao eixo horizontal; o “Aumento 2” chamado de “Rotação

Figura 19 – Imagem sem aumento de dados.

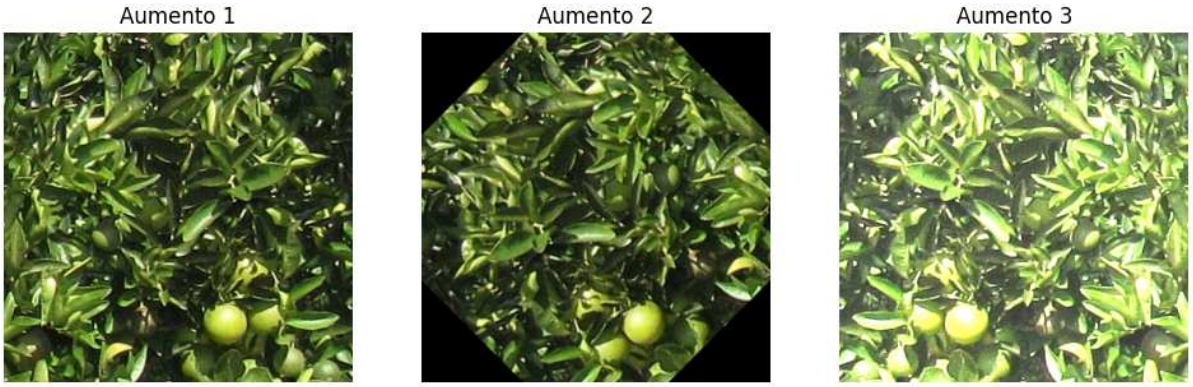


Figura 20 – Imagens geradas por aumento de dados.

Aleatória", é obtido rotacionando a imagem em torno de seu centro; e o "Aumento 3" é o "Ajuste de Brilho e Contraste", nesse caso o brilho e contraste da imagem são aumentados ou diminuídos, de acordo com a configuração dada. Referente às caixas delimitadoras, ao ser efetuado o aumento de dados, as caixas precisam ser adaptadas pelo modelo para delimitarem as novas localizações e tamanhos dos objetos.

No trabalho de Hernández-García e König (2018) vemos que redes treinadas com aumento de dados tem melhor desempenho para se adaptar a diferentes dados.

3.7 Transferência de Aprendizado

Transferência de Aprendizado é a técnica que proporciona que um modelo A já treinado possa melhorar a generalização de outro modelo B. Na transferência de aprendizado assumimos que muitos dos fatores que compõem as variações de A podem ser utilizados para compor as variações de B (GOODFELLOW; BENGIO; COURVILLE, 2016).

No contexto de aprendizado supervisionado podemos, por exemplo, utilizar um modelo treinado com um conjunto relativo a um animal e utilizá-lo para aprimorar a generalização de outro modelo que utiliza um conjunto de outro animal. Mesmo que

os animais presentes em cada conjunto sejam diferentes, as características como sombra, variações geométricas, iluminação, quando transferidas de outro conjunto de pesos já treinados ajudam a aprimorar a aproximação do conjunto a ser treinado.

Na prática é utilizada uma rede base pré-treinada e suas camadas finais de classificação são substituídas pelas camadas do modelo a ser treinado. Deste modo, as camadas pertencentes ao modelo base são "congeladas"e não são treinadas. Essas informações são exemplificadas na prática na subseção 5.1.2 e subseção 5.1.3.

4 Arquiteturas de Redes Profundas

Este capítulo apresentará duas arquiteturas de redes profundas, a *VGG-16* e a *MobileNet*. Ambos são modelos de classificação de imagens e são componentes essenciais para a rede *SSD*, que originalmente utilizava a rede *VGG-16* mas após avanço da tecnologia começou a utilizar também a rede *MobileNet*. Nesse capítulo, será apresentado a estrutura, motivação para o desenvolvimento, e ainda, as diferenças entre as duas arquiteturas apresentadas.

4.1 VGG-16

A arquitetura VGG-16, desenvolvida por Simonyan e Zisserman (2015), é uma rede neural convolucional notável que ganhou reconhecimento por seu desempenho em tarefas de visão computacional, como classificação de imagens. Ela é conhecida por seu foco na profundidade da rede (número de camadas) e no uso de filtros convolucionais 3×3. Abaixo seguem algumas características da rede:

Profundidade da Rede: A VGG-16 é uma rede convolucional profunda com 16 camadas com parâmetros ajustáveis. Ela possui 13 camadas convolucionais e 3 camadas densas (totalizando 16 camadas), como visto na Figura 21. A investigação sobre a profundidade da rede era uma das principais motivações para o desenvolvimento dessa arquitetura.

Filtros Convolucionais 3×3: A VGG-16 utiliza filtros convolucionais de tamanho 3×3 em todas as suas camadas convolucionais. Isso é notável porque, para convoluções convencionais (uma convolução somente), um filtro 3×3 é o menor tamanho que pode capturar todas as noções espaciais em uma imagem. O uso consistente desses filtros ajuda a manter a representação espacial em todo o modelo. Na Figura 22, a notação 3×3 indica a dimensão do filtro convolucional; $64, 128, 256, 512$ o número de filtros convolucionais e *size*, representa o tamanho da imagem de saída dessa camada.

Padding: Todas as camadas convolucionais têm um *stride* (passo) igual a 1 e *padding* (preenchimento) *same*. O *stride* 1 significa que os filtros percorrem a imagem um pixel de cada vez, mantendo a resolução espacial. O *padding same* mantém o tamanho das imagens após a convolução.

Camadas de *Pooling*: A VGG-16 utiliza camadas de *pooling* com dimensões 2×2 e um *stride* de 2. Isso significa que, após cada camada de *pooling*, o mapa de características é reduzido pela metade em ambas as dimensões. Isso ajuda a reduzir a dimensionalidade do mapa de características e torna a computação mais eficiente.

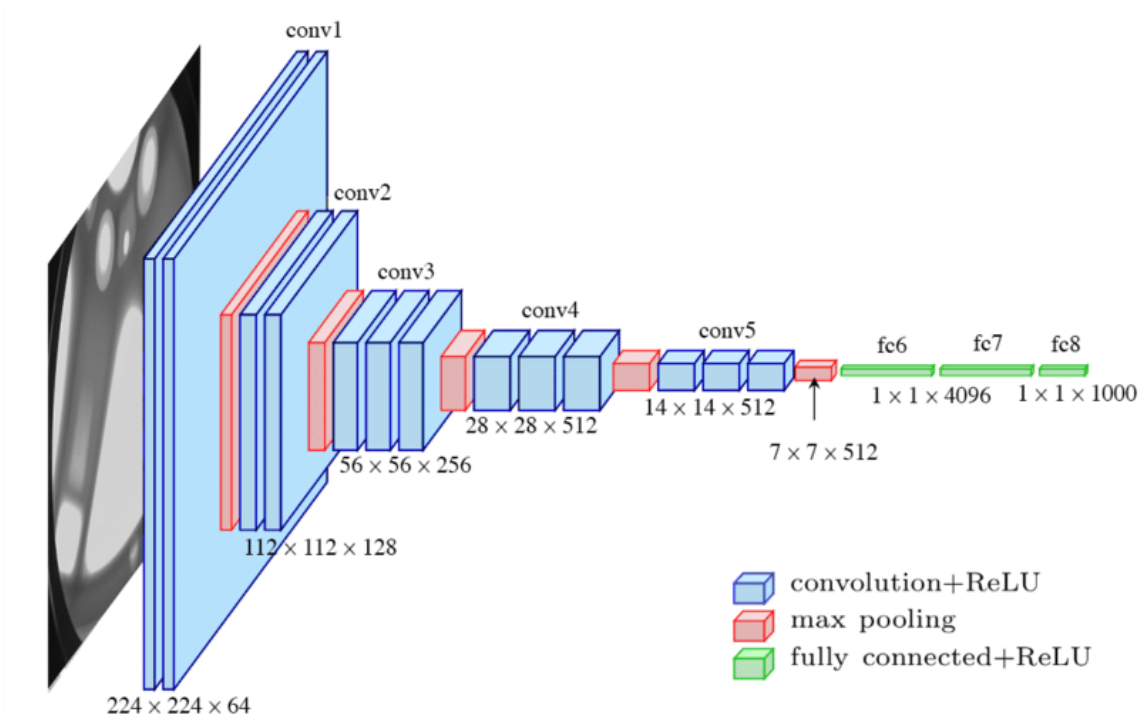


Figura 21 – Ilustração da Rede VGG-16. Fonte: (AK et al., 2017)

Figura 22 – Arquitetura da rede *VGG*. Fonte: (AHMED et al., 2019)

Número de Parâmetros: O uso consistente de filtros 3×3 em vez de filtros maiores tem a vantagem de reduzir o número de parâmetros na rede. De fato, por um lado, uma sequência de 3 convoluções 3×3 resulta em $27c$ parâmetros por filtro em que c é o número de canais. Por outro lado, uma única convolução 7×7 resultaria em $49c$ parâmetros por filtro. Isso torna a rede mais eficiente em termos de computação e menos suscetível ao sobre-ajuste.

A VGG-16 demonstrou a importância da profundidade da rede e do uso de filtros 3×3 na obtenção de bons resultados em tarefas de classificação de imagens.

4.2 *MobileNet v2*

MobileNet é uma série de redes neurais convolucionais desenvolvida para dispositivos móveis. A primeira versão, chamada de "MobileNet V1" (MB1), foi desenvolvida por Howard et al. (2017). A MobileNet consiste em dois hiper-parâmetros que têm como objetivo adequar a rede ao menor tamanho necessário para resolver o problema proposto. Os dois estão presentes na convolução chamada "Convolução separável em profundidade" (CSP), descrita na seção 2.4.

A estrutura da MB1, melhor observada na Tabela 2, é construída sobre CSP, exceto primeira camada. Todas suas camadas são seguidas de BN, vista na seção 2.6, e também da função de ativação *ReLU* descrita na Equação 2.1. Na Figura 23 é possível notar a estrutura do CSP, que a cada convolução é seguido de BN e da função de ativação.

Na Tabela 2, a primeira coluna representa o tipo de camada, a segunda coluna representa a dimensão do filtro convolucional. A terceira representa a dimensão da imagem de entrada da camada. Especificamente na terceira coluna também há a saída da camada observando a entrada referente à próxima camada.

Por conta da CSP, é possível adicionar dois hiper-parâmetros: os multiplicadores de largura e resolução, denotados respectivamente por ι e ν. O multiplicador $\iota \in (0, 1]$ é utilizado para construir modelos para problemas que necessitam de múltiplas camadas mas com menor custo computacional. O parâmetro $\nu \in (0, 1]$ reduz a dimensão da imagem de entrada e a representação de cada camada. Os dois hiper-parâmetros devem ser escolhidos de modo que seus dos produtos com a dimensão da imagem ou do filtro resulte em um número inteiro (HOWARD et al., 2017). Assim, o custo computacional se torna:

$$H_i \times W_i \times \iota M \times \nu D_f \times \nu D_f + \iota M \times \iota N \times \nu D_f \times \nu D_f, \iota \in (0, 1] \text{ e } \nu \in (o, 1].$$

A "Mobilenet V2" (MB2), desenvolvida por Sandler et al. (2018), é uma rede baseada na MB1. Porém, com estrutura de blocos residuais, vistos na seção 2.5. Nessa versão da Mobilenet, a convolução fatorada é expandida para uma convolução de "gargalo invertido", como visto na subseção 2.5.1. Logo, segue estrutura similar de MB1, porém as camadas de CSP são substituídas pelo bloco de gargalo invertido.. É possível observar na Tabela 3 que a diferença entre a primeira e segunda versão da Mobilenet são os blocos de gargalo invertido.

Tabela 2 – Arquitetura da rede Mobilenet v1.

Tipo	**Filtro**	**Entrada**
Convolução Padrão	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Convolução Profunda	$3 \times 3 \times 32$	$112 \times 112 \times 32$
Convolução Pontual	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Convolução Profunda	$3 \times 3 \times 64$	$112 \times 112 \times 64$
Convolução Pontual	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Convolução Profunda	$3 \times 3 \times 128$	$56 \times 56 \times 128$
Convolução Pontual	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Convolução Profunda	$3 \times 3 \times 128$	$56 \times 56 \times 128$
Convolução Pontual	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Convolução Profunda	$3 \times 3 \times 256$	$28 \times 28 \times 256$
Convolução Pontual	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Convolução Profunda	$3 \times 3 \times 256$	$28 \times 28 \times 256$
Convolução Pontual	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
Convolução Profunda Convolução Pontual $\times 5$	$3 \times 3 \times 512$ $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Convolução Profunda	$3 \times 3 \times 512$	$14 \times 14 \times 512$
Convolução Pontual	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Convolução Profunda	$3 \times 3 \times 1024$	$7 \times 7 \times 1024$
Convolução Pontual	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Average *pooling*	7×7	$7 \times 7 \times 1024$
Camada Densa	1024×1000	$1 \times 1 \times 1024$
Softmax	Classificador	$1 \times 1 \times 1000$

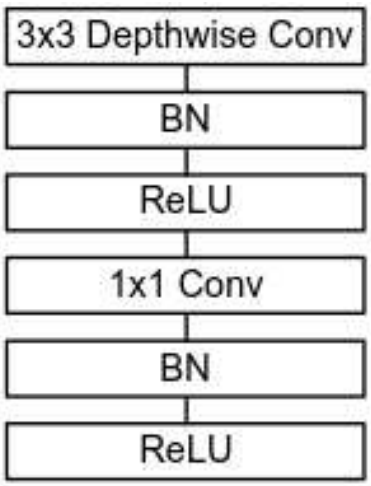


Figura 23 – Convolução Profunda Separada seguida de Normalização em Lote e a função ReLU. Fonte: (HOWARD et al., 2017).

Tabela 3 – Arquitetura da rede Mobilenet $v2$.

Tipo	**Fator de Expansão**	**Filtros**	**Entrada**
Convolução Padrão	-	32	$224 \times 224 \times 3$
Gargalo Invertido	1	16	$112 \times 112 \times 32$
Gargalo Invertido $2\times$	6	24	$112 \times 112 \times 16$
Gargalo Invertido $3\times$	6	32	$56 \times 56 \times 24$
Gargalo Invertido $4\times$	6	64	$28 \times 28 \times 32$
Gargalo Invertido $3\times$	6	96	$14 \times 14 \times 64$
Gargalo Invertido	6	160	$14 \times 14 \times 96$
Gargalo Invertido	6	320	$7 \times 7 \times 160$
Convolução Padrão 1×1	-	1280	$7 \times 7 \times 320$
Average *pooling* 7×7	-	-	$7 \times 7 \times 1280$
Convolução Padrão 1×1	-	1000	$1 \times 1 \times 1280$

5 Detecção de Objetos e a Rede SSD

A detecção de objetos é uma área de visão computacional que consiste em reportar quais objetos estão presentes em uma imagem e fornecer suas localizações. Geralmente, o final desse processo é uma anotação em torno do objeto, o que se dá o nome de "caixa delimitadora" (em inglês, *bounding boxes*).

Uma técnica de detecção de objetos pode ser divida em três estágios (KAUR; SINGH, 2022):

1. Seleção de região;
2. Extração de características;
3. Classificação.

Existem diversos modelos para extração de características, como desenvolvido por Viola e Jones (2004), Lowe (1999). Devido às diversas condições de uma imagem, não é uma tarefa simples desenvolver uma ferramenta robusta para descrever todos os objetos de uma imagem. Por isso, houve um ganho altíssimo com o desenvolvimento das redes neurais convolucionais. As CNNs aprendem características mais complexas e se ajustam automaticamente. Atualmente, existem dois tipos de CNN para detecção de objetos:

1. Redes Baseadas em Regiões: A rede convolucional utiliza duas etapas para detectar um objeto. A primeira etapa examina toda a imagem e a segunda etapa busca sobre as regiões de interesse. O modelo R-CNN (do inglês: *Region-Based Convolutional Neural Networks*) (GIRSHICK et al., 2014) é uma rede neural convolucional baseada em regiões e obteve precisão média de $53,3\%$ sobre o banco de dados PASCALVOC2012 (EVERINGHAM et al., 2012). Esse valor é 30% maior que o melhor resultado de redes anteriores à R-CNN. Redes baseadas em regiões como a R-CNN atuam em duas etapas de treinamento para detectar objetos:

   Geração de Regiões de Interesse (*Region Proposal*): Nesta etapa, o modelo examina a imagem completa para identificar regiões que são potenciais candidatas para conter objetos de interesse. Essas regiões são propostas com base em algoritmos de seleção, e aproximadamente 2000 regiões por imagem são escolhidas.

   Extração de Características e Classificação: Cada uma das regiões propostas é "recortada" da imagem original e redimensionada para uma resolução fixa. Em seguida, um módulo da CNN é aplicado para extrair características relevantes das regiões recortadas. Posteriormente, essas características são usadas para classificar

e localizar objetos dentro das regiões. Cada região é pontuada como positiva ou negativa para cada classe de interesse, identificando assim a presença ou ausência de objetos específicos.

2. Baseada em regressão/classificação. As CNNs baseadas em regressão/classificação podem ser chamadas de CNN de estágio único (em inglês: *Single Shot*). Esses modelos são projetados para realizar a detecção de objetos em um único estágio, o que significa que eles podem mapear diretamente as entradas de imagens para as coordenadas das caixas delimitadoras e as probabilidades das classes em uma única passagem pela rede neural. Isso os torna eficientes em termos de tempo de processamento computacional, pois eliminam a necessidade de etapas intermediárias, como a geração de regiões de interesse, que eram comuns em abordagens anteriores, como o modelo R-CNN. Os modelos de estágio único mais relevantes são os da série YOLO, proposta inicialmente por Redmon et al. (2016), e a SSD (LIU et al., 2016). Em vez de propor regiões e classificá-las depois, esses métodos produzem as coordenadas e classes com apenas uma etapa.

   A SSD e a YOLO são as redes com menor tempo de processamento por imagem quando são treinadas utilizando o banco de dados VOC2007 (EVERINGHAM et al., 2010). A vantagem de redes baseadas em regressão é a rapidez comparada às redes baseadas em regiões.

Nas seções seguintes serão apresentada a rede SSD e a rede SSDLite.

5.1 Rede SSD

Proposta por Liu et al. (2016), a rede SSD é baseada em regressão/classificação. A SSD utiliza uma única CNN para detectar e classificar objetos em uma imagem.

Antes de apresentar a SSD, sua arquitetura e os estágios de treinamento, apresentaremos brevemente os modelos anteriores ao SSD que contribuíram para o desenvolvimento do modelo.

5.1.1 Modelos precursores

Multibox (ERHAN et al., 2014). É um detector baseado em caixas delimitadoras com as seguintes características:

- **Caixas Delimitadoras:** O Multibox gera várias caixas delimitadoras (Bb, em inglês: *bounding boxes*) para cada região de uma imagem. Essas caixas representam possíveis localizações de objetos.

- **Valores de Confiança:** Para cada caixa delimitadora, o Multibox atribui um valor de confiança que indica a probabilidade da caixa conter um objeto de interesse.
- **Treinamento:** Durante o treinamento, a rede é otimizada para prever as caixas delimitadoras e os valores de confiança. O objetivo é fazer com que as caixas com maior valor de confiança se aproximem das caixas padrão (ou *ground truth*) que são fornecidas no conjunto de treinamento.
- **Caixas Padrão (Gt, em inglês: *Ground Truth Boxes*):** As caixas padrão são as caixas delimitadoras reais que envolvem os objetos de interesse nas imagens de treinamento. O objetivo é fazer com que as caixas geradas pelo modelo se aproximem o máximo possível das caixas padrão.

Faster R-CNN (REN et al., 2015). É uma rede baseada em regiões. É composta por 2 módulos: O módulo RPN (*Region Proposal Networks*) e o segundo módulo, o detector. O RPN transforma uma imagem de entrada em um conjunto de regiões retangulares, cada uma com seu valor de confiança.

You Only Look Once (REDMON et al., 2016). É uma rede convolucional baseada em regressão. A sua base é dividir uma imagem em uma "grade". Abaixo são listadas as principais características desta rede:

- **Divisão em Grade:** A imagem de entrada é dividida em uma grade retangular. Cada célula da grade é responsável por fazer previsões sobre objetos que estão contidos dentro dela.
- **Previsões de Caixas Delimitadoras:** Cada célula da grade gera previsões de caixas delimitadoras que descrevem a localização dos objetos. Cada previsão inclui coordenadas "x" e "y" da caixa delimitadora, sua largura e altura.
- **Valores de Confiança:** Além das caixas delimitadoras, cada célula também prevê um valor de confiança que representa a probabilidade de que um objeto esteja presente naquela região da imagem.
- **Arquitetura da Rede:** O modelo YOLO é composto por 24 camadas convolucionais e 2 camadas densas (totalizando 26 camadas). Essas camadas são responsáveis por extrair características da imagem e realizar as previsões de caixas delimitadoras e confiança.
- **Eficiência e Velocidade:** A rede YOLO é conhecida por sua eficiência e velocidade em comparação com abordagens anteriores. Ela é capaz de realizar a detecção de objetos em tempo real, tornando-a adequada para uma ampla gama de aplicações, incluindo vigilância por vídeo, automóveis autônomos e muito mais.

A SSD utiliza técnicas inspiradas nos modelos anteriores. Principalmente pela dificuldade da YOLO de detectar objetos pequenos, a SSD utiliza as caixas delimitadoras de Multibox e Faster R-CNN, como veremos nas próximas sub-seções.

5.1.2 Arquitetura do Modelo

O modelo SSD é, de forma geral, dividido em duas partes: um modelo base, que são as primeiras camadas, e as camadas SSD, que são as últimas camadas do modelo.

5.1.2.1 Modelo Base

O modelo base (em inglês: *Backbone model*) (AMJOUD; AMROUCH, 2020) é o modelo utilizado para extrair características das imagens. Em geral, podem ser utilizados diferentes tipos de rede base na SSD. Os desenvolvedores da SSD, porém, utilizaram a rede *VGG-16* (SIMONYAN; ZISSERMAN, 2015), que representava o estado-da-arte em CNNs para classificação. No caso da SSD, a maioria das camadas pertencem à rede base, e, portanto, a rede base influencia em grande parte no desempenho final do modelo.

Algumas adaptações são necessárias para acoplar o modelo base para uma tarefa de detecção na rede SSD:

1. As camadas densas fc6 e fc7 (veja na Figura 21) são convertidas em camadas convolucionais 3×3 e 1×1;
2. A partir de fc7 iniciam-se as camadas adicionais da SSD.

5.1.2.2 Camadas SSD

O SSD contém 6 camadas convolucionais com *kernel* 3×3 para extração de características, como pode ser visto na Figura 24, descritas como "*Extra Feature Layers*" pelos autores. As camadas convolucionais contribuem para a detecção de objetos fornecendo tanto o valor de confiança para uma classe como o tamanho da caixa delimitadora. Para cada uma dessas camadas um número fixo de convoluções é aplicado.

5.1.2.3 Caixas Delimitadoras

Caixas delimitadoras são regiões retangulares pré-determinadas por meio da configuração do modelo, semelhante às ancoras desenvolvidas por Ren et al. (2015). A cada localização em cada mapa de características, o modelo prevê 4 outras regiões próximas. Cada mapa de características contém um conjunto de caixas delimitadoras. O modelo aplica um filtro convolucional para cada caixa. Logo, são aplicadas várias convoluções em dimensões diferentes.

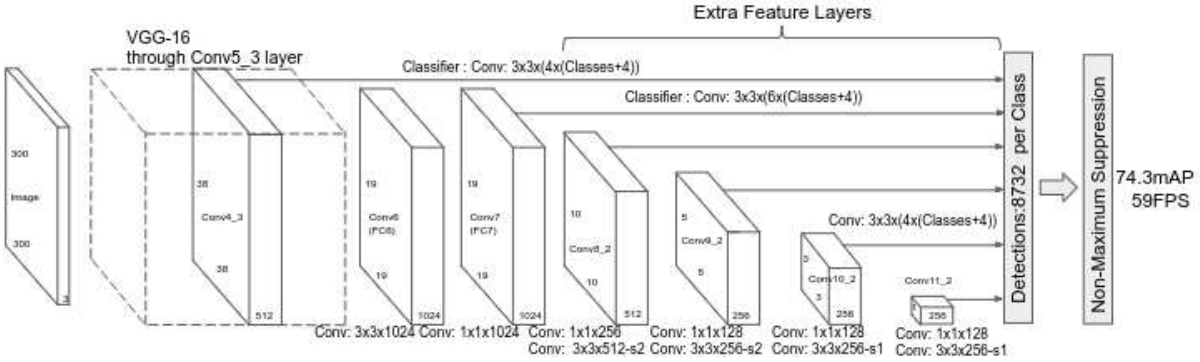


Figura 24 – Arquitetura da rede SSD. Fonte: (LIU et al., 2016)

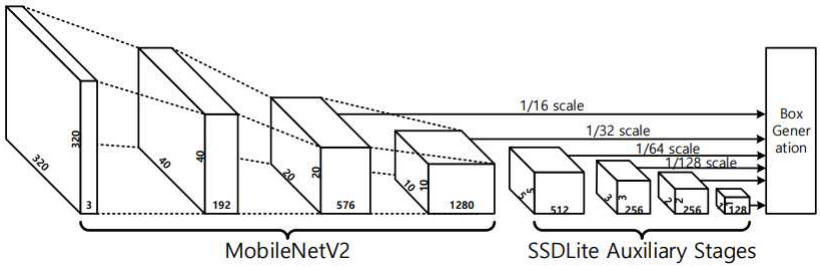


Figura 25 – Arquitetura da rede SSDLite. Fonte: (KANG, 2023)

Especificamente, para cada caixa, o modelo computa c valores de classes. Assim, para k caixas delimitadoras são $(c+4)k$ convoluções em um mapa de características e $(c+4)kmn$ convoluções são aplicadas por mapa.

Durante o treinamento, essas caixas são associadas às caixas padrão, que são regiões retangulares pré-definidas pelo banco de dados, onde apontam os objetos a serem detectados. Ainda durante o treinamento, as possíveis detecções estão nessas caixas delimitadoras.

5.1.3 SSDlite

SSDlite é uma adaptação da SSD que foi apresentada por Sandler et al. (2018). Na Figura 25 vemos quea diferença da SSDLite para SSD é a rede base. Por exigir baixo custo computacional, a SSDLite pode ser utilizada em softwares embarcados, dispositivos móveis, inclusive, o experimento relatado nessa dissertação utiliza a rede SSDlite. A SSDlite adapta a rede SSD utilizando a rede MobileNet como sua rede base e também substitui suas convoluções auxiliares por convoluções separáveis em profundidade.

Na SSD as camadas densas da rede VGG-16 são substituídas por camadas convolucionais com *kernel* 3×3, aumentando a complexidade da rede. Na SSDLite, as camadas densas são removidas. Assim, a rede SSD pode ter maior poder de detecção, mas conta com maior complexidade computacional.

As camadas convolucionais da SSD são substituídas por camadas separadas na SSDlite. As camadas auxiliares da SSDlite consistem cada uma de uma convolução 1×1 que reduz canais, uma convolução 3×3 profunda e uma convolução pontual 1×1 que recupera os canais. Assim, as camadas auxiliares são compostas por blocos de gargalo.

A seção a seguir descreve o treinamento da rede SSD, que é o mesmo para a SSDlite.

5.1.4 Treinamento

O treinamento determina qual caixa delimitadora detectada (Dt) corresponde a uma caixa padrão (Gt), e ajusta os pesos da rede de acordo com as detecções na caixa. Assim, cada caixa padrão é associada a uma saída. Como o conjunto de saídas é limitado, o trabalho de treinamento é associar caixas padrão a caixas delimitadoras detectadas. Após isso, a função perda e os algoritmos de otimização são aplicados.

Nesta seção será apresentada a estratégia de associação, o objetivo do treinamento e as funções perda que são aplicadas para ajuste dos pesos.

5.1.4.1 Estratégia de Associação

Para iniciar o treinamento, o modelo associa as caixas padrão (Gt) às caixas delimitadoras (Bb) utilizando o IOU. Especificamente, as caixas delimitadoras (Bb) utilizadas para o treinamento são tais que $IOU \geq 0,5$. Desse modo, haverá somente caixas delimitadoras que estão localizadas próximas ou sobre as caixas padrão.

Para aprender objetos de escalas diferentes, a SSD processa a imagem em escalas diferentes. Isso causa um problema de proporção entre as caixas padrão e os mapas de características, pois camadas diferentes têm filtros de tamanhos diferentes. Assim, as caixas são adaptadas para cada camada.

Suponha m camadas, $S_{min} = 0,2$ (escala da menor camada em relação à primeira), $S_{max} = 0,9$ (escala da maior camada em relação a primeira). O modelo calcula a escala S^k de cada caixa delimitadora da camada k através da equação a seguir (LIU et al., 2016):

$$S^k = S_{min} + \frac{S_{max} - S_{min}}{m - 1}(k - 1) \tag{5.1}$$

A SSD gera 4 novas caixas delimitadoras (deslocamentos) em relação à original. A proporção de uma caixa adicional é: $Bb_r \in \{1; 2; 3; \frac{1}{2}; \frac{1}{3}\}$. Assim, a altura ($h$), largura ($w$), são dadas por:

$$w_k^r = S_k\sqrt{Bb_r}; \tag{5.2}$$

e

$$h_k^r = \frac{S_k}{\sqrt{Bb_r}}. \tag{5.3}$$

Seja f_k o tamanho do k-ésimo mapa, (i_r, j_r) o centro original da caixa, o centro da caixa deslocado, $\overline{o}_k^r$, é dado por:

$$\overline{o}_k^r = \left(\frac{i_r + 0,5}{f_k}, \frac{j_r + 0,5}{f_k} \right). \tag{5.4}$$

Com essas informações a SSD pode prever caixas em escalas e proporções de todas as localizações em mapas diferentes.

5.1.4.2 Objetivo do Treinamento

O objetivo do treinamento é prever blocos delimitadores e seus valores de confiança de modo que os maiores valores de confiança estejam bem associados à devida caixa padrão. Assim, os parâmetros da rede são ajustados de acordo com a associação entre as caixas delimitadoras e padrão.

Uma das principais diferenças do treinamento da rede SSD é encontrar caixas delimitadoras a partir das caixas padrão. Logo, cada Gt precisa estar associado a um grupo de saídas para o treinamento. Dessa forma, o treinamento é mais econômico e rápido, minimizando a quantidade de caixas delimitadoras com valores de confiança baixos (ERHAN et al., 2014).

5.1.4.3 Funções Custo da SSD

As funções custo do treinamento da rede SSD têm como objetivo mensurar o desempenho das detecções em relação à posição das caixas detectadas e os valores de confiança. A função perda é a soma de duas funções chamadas *"localization loss"* e *"confidence loss"*. Assim, o treinamento é formulado para otimizar não somente o valor de confiança de uma saída, mas também a localização da caixa detectada.

Sejam l os parâmetros da caixa delimitadora detectada (Dt), c a classe do objeto em questão, g os parâmetros da caixa padrão (Gt) e x o valor de confiança obtido, N o número de caixas detectadas. A soma ponderada entre a *localization loss* (loc) e a *confidence loss* $(conf)$, $\alpha = 1$ determinado por validação cruzada por Liu et al. (2016), é dada por:

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)), \tag{5.5}$$

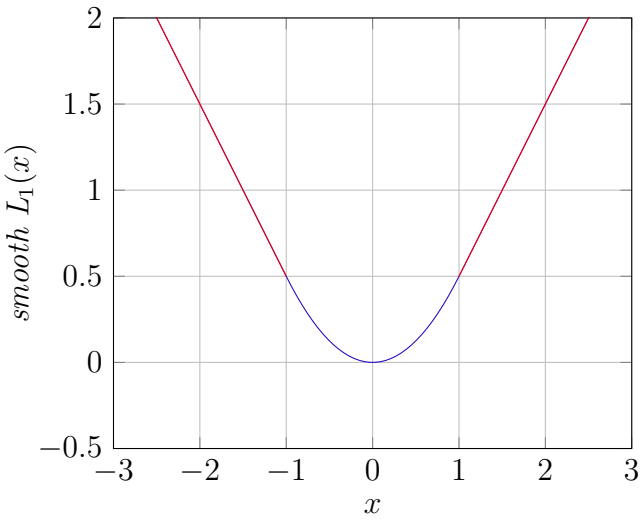
com $L(x, c, l, g) = 0$, quando $N = 0$ (LIU et al., 2016).

Nos dois parágrafos a seguir apresentaremos cada uma das funções custo de localização e confiança.

1. *Localization Loss*: A *localization loss* é baseada na função *Smooth*-L_1 entre os parâmetros (l) da caixa prevista (Dt) e os parâmetros (g) da caixa padrão (Gt). A

função *Smooth-*L_1 executa uma regressão para prever a localização das caixas. A vantagem da função *Smooth-*L_1 está em ser menos sensível a *outliers* (GIRSHICK, 2015; FU; SHVETS; BERG, 2019). Ela é dada por:

$$\textit{smooth-}L_1(x) = \begin{cases} \dfrac{0.5x^2}{\beta}, & \text{se } |x| < \beta, \\ |x| - 0.5\beta, & \text{caso contrário.} \end{cases} \tag{5.6}$$



O hiper-parâmetro β determina em que momento a função *smooth* L_1 se torna linear. Ele também determina o quão sensível a *outliers* será a função perda. A divisão entre as funções ocorre para diminuir a ação de *outliers* no resultado, pois a função $L_1(x) = |x| - 0,5\beta$ não é tão afetada por *outliers*.

A regressão feita usando a *Smooth-*L_1 é usada para ajustar os centros (cx, cy), largura (w) e altura (h) da caixa delimitadora prevista (Dt) com a caixa padrão (Gt).

Os valores de posição das caixas serão indexados $m \in cx, cy, w, h$, em que cx e cy representam as coordenadas do centro e w e h representam a largura e altura da caixa. Pos é o conjunto das i caixas detectadas positivas e $\hat{g}_j^m$ a distância relativa entre a j-ésima caixa padrão Gt e a caixa delimitadora fixada pelo próprio modelo Bb que foi associada à caixa detectada, $X_{ij}^c \in 0, 1$ indica a relação entre a i-ésima caixa delimitadora e a j-ésima caixa padrão para a classe c. Temos que L_{loc} é dada por:

$$L_{loc}(x, l, g) = \sum_{i \in Pos} \sum_{m \in cx,cy,w,h} X_{ij}^c \; smoothL_1(l_i^m - \hat{g}_j^m), \tag{5.7}$$

onde l_i^m é referente às coordenadas da caixa prevista. (GIRSHICK, 2015). Para g_j^m coordenadas das caixas padrão e d_i^m as coordenadas das caixas delimitadoras as

transformações são, como visto em Liu et al. (2016), dado por:

$$\hat{g}_j^{cx} = \frac{g_j^{cx} - d_i^{cx}}{d_i^w} \tag{5.8}$$

$$\hat{g}_j^{cy} = \frac{g_j^{cy} - d_i^{cy}}{d_i^h}, \tag{5.9}$$

$$\hat{g}_j^{w} = \log \frac{g_j^{w}}{d_i^w}, \tag{5.10}$$

$$\hat{g}_j^{h} = \log \frac{g_j^{h}}{d_i^h}. \tag{5.11}$$

As expressões acima são utilizadas na Equação 5.7 para calcular o custo das detecções. Deste modo a função custo avalia a regressão para todas as coordenadas de todas as caixas detectadas.

2. *Confidence Loss*: A função *Confidence Loss* é uma versão adaptada da entropia cruzada binária sobre as c classes, que relaciona as caixas delimitadoras positivas com negativas, tais que $IoU < 0,5$. A função *Softmax* calcula um valor de confiança ς que pode ser também interpretado como a probabilidade para cada classe. Especificamente, ς_i^c é o valor de confiança da i-ésima caixa detectada em relação à classe c dado por:

$$\hat{\varsigma}_i^c = \frac{exp(\varsigma_i^c)}{\sum_c exp(\varsigma_i^c)}, \tag{5.12}$$

em que ς_i^c é uma saída associada à i-ésima caixa delimitadora.

Sejam Pos e Neg o conjunto das caixas detectadas que são pertencentes a classe e não pertencentes a alguma classe, respectivamente. L_{conf} é dado por:

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} X_{ij}^c \log(\hat{\varsigma}_i^c) - \sum_{i \in Neg} \log(\hat{\varsigma}_i^c). \tag{5.13}$$

Concluindo, vimos como a SSD realiza seu treinamento, que é o mesmo da SSDLite, no próximo capítulo podemos analisar os resultados do experimento do trabalho.

6 Metodologia, Resultados e Discussão

Neste capítulo serão apresentados os resultados da aplicação da rede SSDLite ao banco de dados disponibilizado pelo Fundecitrus e processado pela Embrapa. O objetivo é analisar os resultados e o desempenho da rede SSDLite, e também realizar a comparação com o desempenho da rede SSD.

De início veremos a metodologia do trabalho. A segunda seção apresenta a técnica de equalização de histograma utilizada para o pré-processamento das imagens. E em seguida comentamos o banco de dados considerado e como ele foi manipulado para ser adaptado às configurações computacionais do modelo. Na seção seguinte apresentamos as configurações determinadas para o treinamento do modelo, como número de iterações e hiper-parâmetros. Na penúltima seção serão analisados os resultados obtidos, como o desempenho do modelo. Por fim, veremos a comparação dos modelos SSDLite e SSD.

6.1 Metodologia

O fluxo do trabalho inicia-se pela base da dados, fornecida pelo Fundecitrus e processada pela Embrapa, o processo realizado foi de recortar as imagens, pois eram imagens de árvores completas, e também de realizar as anotações dos rótulos que indicam a posição dos frutos, ao fim obtemos 3028 imagens (416×416) de frutos em laranjeiras.

Durante o pré-processamento realizado pelo autor, foram aplicados as divisões no banco de dados em três conjuntos: treino (70%), validação (15%) e teste (15%). Além disso todas as imagens são transformadas utilizando Equalização de Histograma.

O treinamento da rede neural convolucional SSDLite foi realizado na plataforma *Google Colab* com a GPU *NVIDIA L4 Tensor Core* e todo o código para execução é fornecido pelo *TensorFlow* (ABADI et al., 2015).

O critério de parada antecipada foi dado pelo não diminuição da função custo por 10 épocas consecutivas. Após a parada antecipada o modelo escolhido seria o modelo de treinamento com o maior valor da métrica de precisão considerada. Ao fim, todos os resultados são calculados pelo próprio *TensorFlow*.

6.2 Equalização de Histograma

A equalização de histograma é uma técnica de processamento de imagens utilizada para melhorar o seu contraste. Esse método transforma o histograma original em

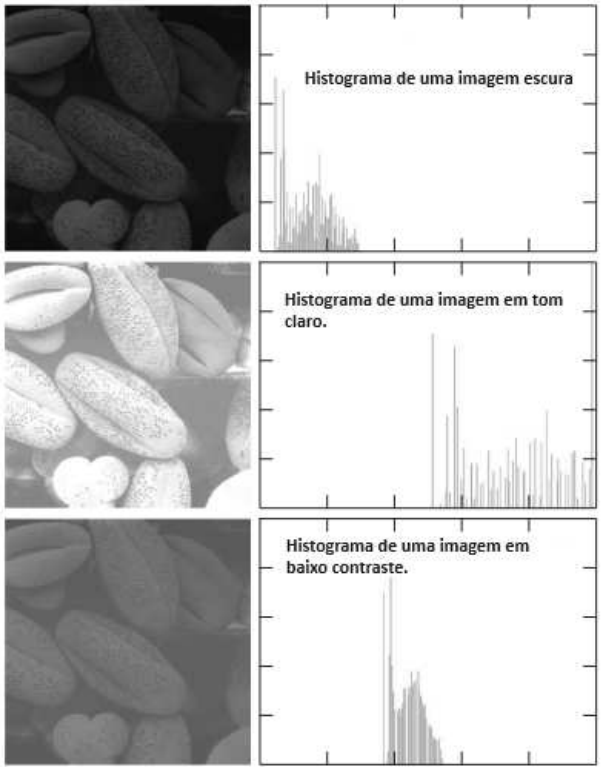


Figura 26 – Imagens à esquerda e seus respectivos histogramas à direita. Fonte: (GONZALEZ; WOODS, 2006), modificado pelo autor.

um histograma uniforme. Isso permite que áreas de baixo contraste local obtenham um contraste mais alto.

O histograma de uma imagem é um gráfico de frequência que apresenta a ocorrência de cada uma das intensidades de pixel. Na Figura 26 temos exemplos de histogramas, e também é possível notar a diferença das posições com maior ocorrência relativo às intensidades.

O histograma é construído utilizando a contagem absoluta ou relativa de cada pixel para dada κ-ésima intensidade. Seja η o número total de pixels na imagem, η_κ o número de pixels para a intensidade κ, π denota as intensidades da imagem a serem processadas, e varia entre 0 e $\kappa - 1$. Ao fim, a probabilidade da κ-ésima intensidade é dada por:

$$\rho_\pi(\pi_\kappa) = \frac{\eta_\kappa}{\eta}.$$

Sabendo disso, para o ajuste do histograma, utiliza-se a função de distribuição acumulada, de modo que:

$$i_\kappa = \sum_{j=0}^{\kappa} \rho_\pi(\pi_j).$$

Após essa distribuição é possível ter um valor uniforme entre todas as intensidades na imagem. Na Figura 27 é possível notar as diferenças de contraste entre uma imagem

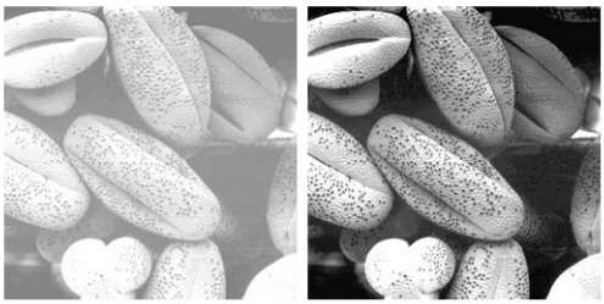


Figura 27 – Imagem sem processamento à esquerda, e imagem processada com equalização de histograma à direita. Fonte: (GONZALEZ; WOODS, 2006).

convencional e uma imagem processada com a equalização de histograma.

6.3 Banco de Dados

A base de dados disponibilizada é composta por 3028 imagens (416×416) de frutos em laranjeiras. Existe uma grande variação de luminosidade, de qualidade de imagem e também de posição dos frutos e tamanhos de laranjas no banco de dados, como podemos ver nas Figura 28, Figura 29 e Figura 30.

Dadas variações de características nas imagens, elas foram pré-processadas utilizando equalização de histograma, melhorando o contraste e aumentando a riqueza dos detalhes, como podemos notar as diferenças na Figura 31 e o seu histograma na e logo abaixo a imagem pré-processada na e seu histograma equalizado. Comparando os dois histogramas é possível notar que o primeiro está indicando tons com intensidade mais escura e o segundo está uniforme.

A base de imagens foi dividida em três conjuntos: treino (70%), validação (15%) e teste (15%).

O modelo SSDLite é configurado para redimensionar as imagens para 300×300 e identificar as anotações no formato *xmin, ymin, xmax, ymax* para as caixas padrão.

6.4 Configurações do Modelo

O modelo foi configurado para treinar ao longo de 40.000 iterações, pois após esse número de iterações não houve diminuição da função custo por 10 épocas consecutivas. Cada iteração corresponde a uma atualização do gradiente. O critério de escolha do modelo foi baseado no maior valor de AP sobre o conjunto de validação. O tamanho do lote foi fixado em 64 imagens, o que significa que a cada iteração o modelo processa um conjunto de 64 imagens.

Figura 28 – Imagem com pouca luminosidade.

Figura 29 – Imagem com frutos obstruídos por folhas.

No treinamento, adotou-se a estratégia de transferência de aprendizado. A rede MobileNet V2 corresponde ao modelo previamente treinado no banco de dados Microsoft COCO (LIN et al., 2014). Foram aplicadas técnicas de regularização, incluindo o uso de dropout e a Regularização de Tikhonov, com um hiperparâmetro igual a 0.00004, para ajuste dos pesos das camadas da SSD.

Os hiper-parâmetros do treinamento foram os mesmos utilizados por Liu et al. (2016), incluindo as proporções das caixas delimitadoras e uma taxa de aprendizado fixada em 0.001. Além disso, foram aplicados métodos de aumento de dados, como"Corte Aleatório" e "Espelhamento Horizontal".

As métricas utilizadas foram as apresentadas pela competição COCO, descritas

Figura 30 – Imagem com menor qualidade por efeito da luminosidade.

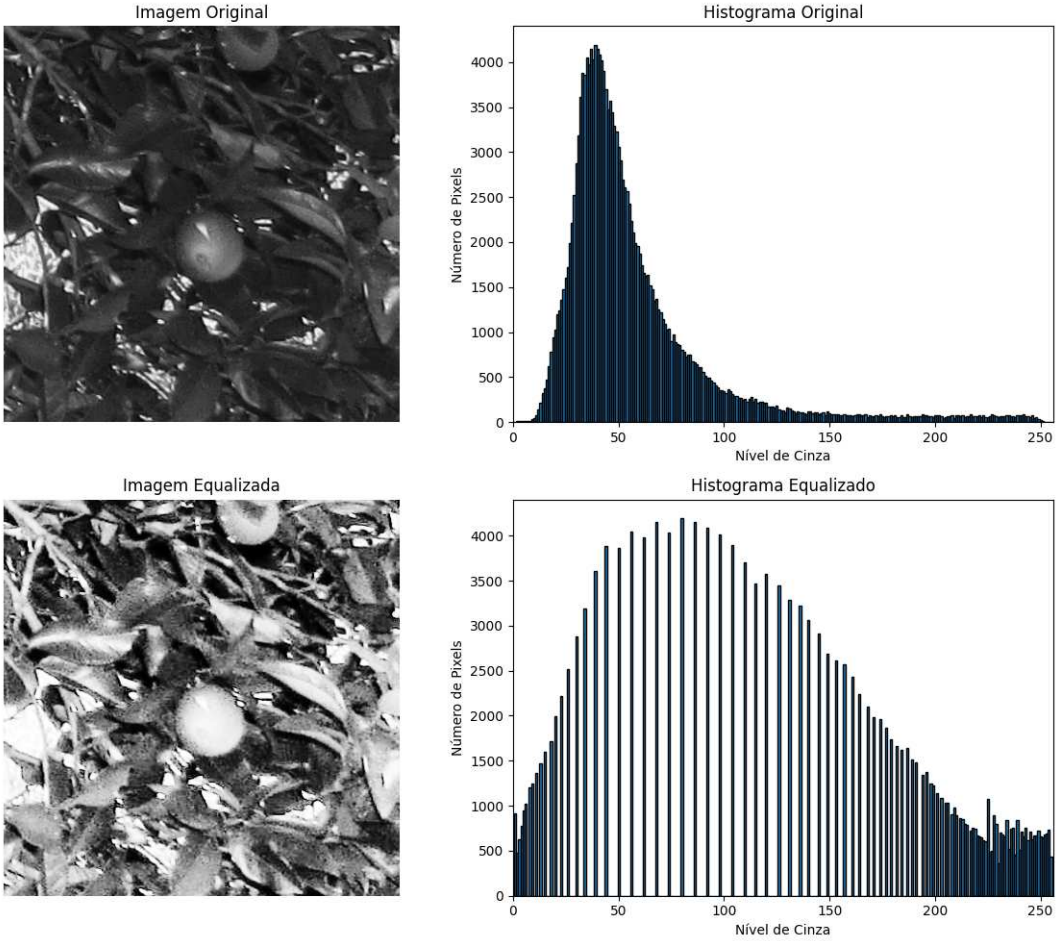


Figura 31 – Imagem original e após a equalização com seus devidos histogramas.

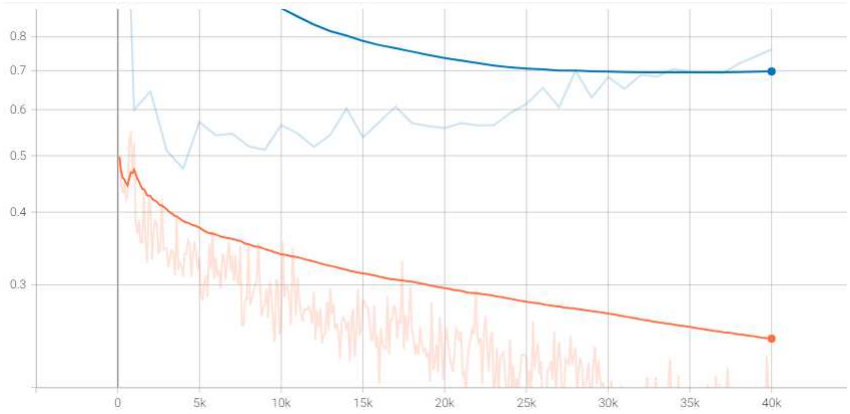


Figura 32 – Curva de custo total do treinamento suavizada pela média móvel exponencial (Em laranja referente ao conjunto de treinamento e em azul referente ao conjunto de validação, em azul claro a curva sem suavização).

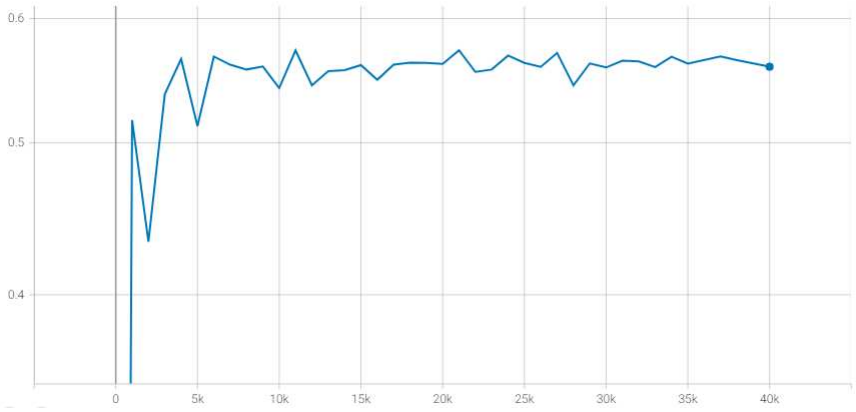


Figura 33 – Curva dos valores de AP.

na seção 3.5.

A escolha do modelo foi dada pelo maior valor de AP no intervalo de treinamento e foi gerado pela iteração de número 21.000, com AP igual a 0.5726, visto na Figura 33.

6.5 Resultados

Nesta seção, apresentaremos os resultados obtidos durante a avaliação do modelo proposto, abordando métricas de desempenho, análise de sobre-ajuste e critério de parada.

1. Desempenho da Rede: O desempenho da rede foi avaliado utilizando diversas métricas, incluindo AP, AP@Small, AP@0.5 e AR, além das curvas de custo total (observada na Figura 32), custo de classificação e custo de localização. A AP para $IOU = 0.5$ é a métrica mais relevante nesse caso, pois o problema para a identificação das laranjas não exige que o modelo tenha valores altos para valores de IOU maiores,

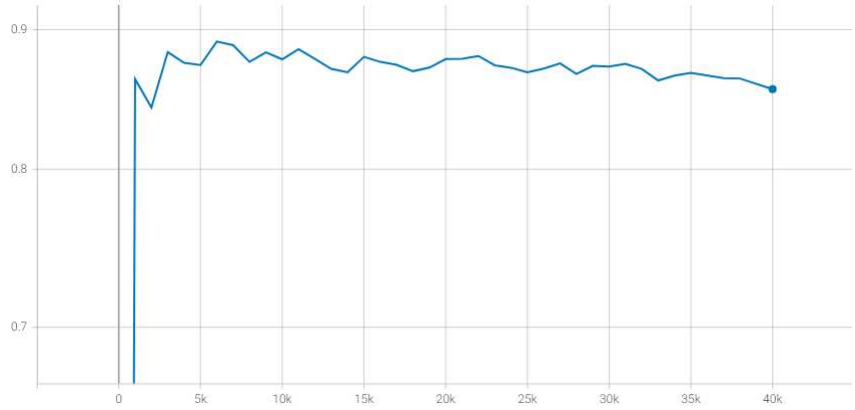


Figura 34 – Curva dos valores de AP para IOU $= 0.50$.

pois somente é necessária a identificação e não a detecção de diversos objetos.

Nesse caso, o modelo escolhido obteve valor de $0,88$ para AP@0.5, que é considerado bom para o modelo. Na Figura 34 notamos que ao longo do treinamento a tendência era o valor abaixar.

2. Análise do Desempenho por Tamanho de Objeto: Durante todo o treinamento, o maior valor para $AP@Small$ que se refere ao valor de AP para objetos pequenos foi de 0.13. Por outro lado, para objetos grandes o valor de $AP@Large$, que se refere ao valor de AP para objetos grandes, não foi menor que 0.60. Isso indica que o desempenho do modelo com objetos pequenos é insuficiente, logo, a SSD com arquitetura Mobilenet v2 pode não ser o melhor detector de objetos para um problema que somente envolve objetos pequenos. Isso provavelmente é causado pelo baixo número de parâmetros da rede, que também é sua vantagem por aumentar a velocidade de treinamento.

3. Avaliação do Modelo:

   O desempenho do modelo é adequado para o problema em questão, especialmente na identificação de objetos de tamanho médio. No entanto, para objetos pequenos e para detecções que requerem maior precisão (valores de IOU mais altos), redes mais recentes podem apresentar um desempenho superior. Mesmo com um número reduzido de parâmetros, o modelo alcançou um bom valor de AP@0.5 igual a 0.88, e um valor de AP superior a 0.5. Nas imagens apresentadas na Figura 35, podemos observar um resultado satisfatório do treinamento da rede.

4. Consumo de memória na inferência: Durante a inferência, cada imagem consumiu cerca de $3mb$ em $2,5$ segundos de execução e também o modelo exige cerca de $0.8B$ Flops, abaixo do modelo da SSD que exige $34.36B$ Flops.

Figura 35 – Imagem com anotações originais.

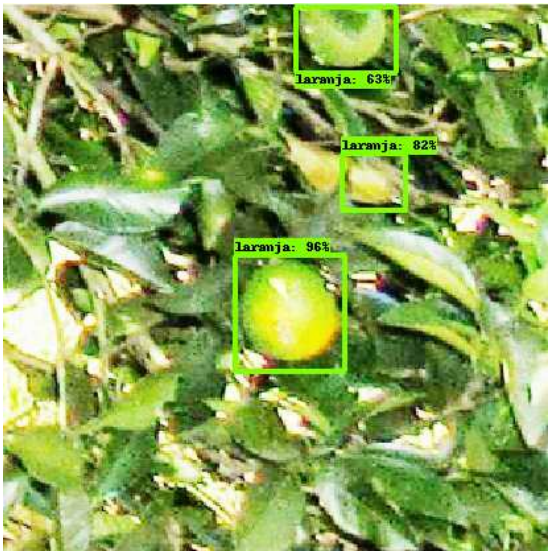


Figura 36 – Imagem com as detecções

	AP@0.5	Parâmetros	Flops
SSD	0.814	$27M$	$34.36B$
SSDLite	0.88	$4M$	$0.8B$

Tabela 4 – SSD x SSDLite

Os resultados obtidos demonstram a eficácia do modelo proposto, especialmente para objetos de tamanho médio e grande. Agora nos resta comparar esses resultados com o desempenho da rede SSD.

6.6 Comparação dos resultados

Na Tabela 4 vemos os resultados comparando a rede SSDLite e a rede SSD para o problema de identificação de laranjas em copas de laranjeiras. É possível notar que a rede SSDLite obteve precisão média para $IOU = 0.5$ maior que a SSD e também com muito menos parâmetros, assim, com custo operacional menor. Logo, a rede SSDLite, mesmo obtendo desempenho inferior em detecção de objetos pequenos, é superior à rede SSD, seja em desempenho, seja em custo computacional.

7 Considerações Finais

A detecção de objetos aplicada à agricultura tem papel essencial para a otimização no trabalho de contagem de frutos.

Neste trabalho, mesmo com um banco de dados com imagens de resolução não tão altas e características diversas nas imagens, foi possível construir um detector com bom desempenho e baixo custo computacional, com desempenho superior ao da rede SSD com o mesmo banco de imagens. E ainda, é possível utilizar o modelo em dispositivos móveis ou ainda realizar o treinamento com outros bancos de dados sem a necessidade de alto processamento ou grande quantidade de memória.

No contexto da citricultura, é necessário obter maneiras de contabilizar o número total de frutos em copas de laranjeiras para assim poder utilizar esses dados para otimizar o trabalho na cultura de laranjas, que tem potencial de otimizar o planejamento para a safra anual.

Referências

ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; GOODFELLOW, I.; HARP, A.; IRVING, G.; ISARD, M.; JIA, Y.; JOZEFOWICZ, R.; KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANé, D.; MONGA, R.; MOORE, S.; MURRAY, D.; OLAH, C.; SCHUSTER, M.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.; TALWAR, K.; TUCKER, P.; VANHOUCKE, V.; VASUDEVAN, V.; VIéGAS, F.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y.; ZHENG, X. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>. Citado na página 58.

AHMED, M.; KIM, Y.; WOO, J.; BASHAR, R.; RHEE, P. Two person interaction recognition based on effective hybrid learning. *KSII Transactions on Internet and Information Systems*, v. 13, 03 2019. Citado 2 vezes nas páginas 10 e 45.

AK, R.; FERGUSON, M.; LEE, Y.-T.; LAW, K. Automatic localization of casting defects with convolutional neural networks. In: . 2017 IEEE International Conference on Big Data (BigData 2017), 2nd Symposium on Data Analytics for, Boston, MA, 2017. Disponível em: <https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=924455>. Citado 2 vezes nas páginas 10 e 45.

AMIN, M. Z.; NADEEM, N. Convolutional neural network: Text classification model for open domain question answering system. *CoRR*, abs/1809.02479, 2018. Disponível em: <http://arxiv.org/abs/1809.02479>. Citado na página 19.

AMJOUD, A. B.; AMROUCH, M. Convolutional neural networks backbones for object detection. In: MOATAZ, A. E.; MAMMASS, D.; MANSOURI, A.; NOUBOUD, F. (Ed.). *Image and Signal Processing*. Cham: Springer International Publishing, 2020. p. 282–289. ISBN 978-3-030-51935-3. Citado na página 52.

AMJOUD, A. B.; AMROUCH, M. Object detection using deep learning, CNNs and vision transformers: A review. *IEEE Access*, v. 11, p. 35479–35516, 2023. Citado na página 16.

BASHA, S. S.; DUBEY, S. R.; PULABAIGARI, V.; MUKHERJEE, S. Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*, Elsevier BV, v. 378, p. 112–119, Feb 2020. ISSN 0925-2312. Disponível em: <http://dx.doi.org/10.1016/j.neucom.2019.10.008>. Citado na página 21.

BOTTOU, L.; CURTIS, F. E.; NOCEDAL, J. Optimization methods for large-scale machine learning. *SIAM Review*, v. 60, n. 2, p. 223–311, 2018. Disponível em: <https://doi.org/10.1137/16M1080173>. Citado na página 33.

CERQUEIRA, L. M.; SOUZA, K. X. S. d.; TERNES, S.; NETO, J. C. Usando a rede neural faster-rcnn para identificar frutos verdes em pomares de laranja. In: *Anais do Congresso Interinstitucional de Iniciação Científica*. Campinas: Embrapa Informática Agropecuária, 2020. p. 1–9. ISBN 978-65-88414-00-2. Embrapa Agricultura Digital, 07/12/2020. Atualizado em 10/12/2020. Disponível em: <https:

//ainfo.cnptia.embrapa.br/digital/bitstream/item/218841/1/RE20605-CIIC-2020.pdf>. Citado na página 17.

DORFLER, M.; BAMMER, R.; GRILL, T. Inside the spectrogram: Convolutional neural networks in audio processing. In: *2017 International Conference on Sampling Theory and Applications (SampTA)*. [S.l.: s.n.], 2017. p. 152–155. Citado na página 19.

EL-AMIR, H.; HAMDY, M. *Deep Learning Pipeline: Building a Deep Learning Model with TensorFlow*. 1st. ed. USA: Apress, 2019. ISBN 1484253485. Citado 2 vezes nas páginas 10 e 26.

ERHAN, D.; SZEGEDY, C.; TOSHEV, A.; ANGUELOV, D. Scalable object detection using deep neural networks. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2014. p. 2155–2162. Citado 2 vezes nas páginas 50 e 55.

EVERINGHAM, M.; GOOL, L. V.; WILLIAMS, C. K. I.; WINN, J.; ZISSERMAN, A. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*. 2010. Http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. Citado na página 50.

_____. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. 2012. Http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. Citado na página 49.

FU, C.; SHVETS, M.; BERG, A. C. *RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free*. 2019. Disponível em: <http://arxiv.org/abs/1901.03353>. Citado na página 56.

GAMA, F.; MARQUES, A. G.; LEUS, G.; RIBEIRO, A. Convolutional neural network architectures for signals supported on graphs. *Trans. Sig. Proc.*, IEEE Press, v. 67, n. 4, p. 1034–1049, feb 2019. ISSN 1053-587X. Disponível em: <https://doi.org/10.1109/TSP.2018.2887403>. Citado na página 19.

GÉRON, A. *Hands-on machine learning with Scikit-Learn and TensorFlow concepts, tools, and techniques to build intelligentsystems*. 1. ed. O'Reilly Media, 2017. Paperback. ISBN 1491962291. Disponível em: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1491962291>. Citado 4 vezes nas páginas 16, 25, 32 e 34.

GIRSHICK, R. Fast r-cnn. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2015. p. 1440–1448. Citado na página 56.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2014. p. 580–587. Citado 2 vezes nas páginas 17 e 49.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: TEH, Y. W.; TITTERINGTON, M. (Ed.). *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010. (Proceedings of Machine Learning Research, v. 9), p. 249–256. Disponível em: <https://proceedings.mlr.press/v9/glorot10a.html>. Citado na página 29.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing (3rd Edition)*. USA: Prentice-Hall, Inc., 2006. ISBN 013168728X. Citado 3 vezes nas páginas 10, 59 e 60.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado 4 vezes nas páginas 23, 33, 41 e 42.

GREMES, M. F.; FERMO, I. R.; KRUMMENAUER, R.; FLORES, F. C.; ANDRADE, C. M. G.; LIMA, O. C. d. M. System of counting green oranges directly from trees using artificial intelligence. *AgriEngineering*, v. 5, n. 4, p. 1813–1831, 2023. ISSN 2624-7402. Disponível em: <https://www.mdpi.com/2624-7402/5/4/111>. Citado na página 18.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 2nd. ed. USA: Prentice Hall PTR, 1998. ISBN 0132733501. Citado na página 21.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. p. 770–778. Citado 3 vezes nas páginas 18, 27 e 29.

HERNÁNDEZ-GARCÍA, A.; KÖNIG, P. Further advantages of data augmentation on convolutional neural networks. In: KŮRKOVÁ, V.; MANOLOPOULOS, Y.; HAMMER, B.; ILIADIS, L.; MAGLOGIANNIS, I. (Ed.). *Artificial Neural Networks and Machine Learning – ICANN 2018*. Cham: Springer International Publishing, 2018. p. 95–103. ISBN 978-3-030-01418-6. Citado na página 42.

HOWARD, A. G.; ZHU, M.; CHEN, B.; KALENICHENKO, D.; WANG, W.; WEYAND, T.; ANDREETTO, M.; ADAM, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. Disponível em: <http://arxiv.org/abs/1704.04861>. Citado 4 vezes nas páginas 10, 28, 46 e 48.

HUBEL, D. H.; WIESEL, T. N. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, v. 148, n. 3, p. 574–591, 1959. Disponível em: <https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1959.sp006308>. Citado na página 24.

IOFFE, S.; SZEGEDY, C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. [S.l.]: JMLR.org, 2015. (ICML'15), p. 448–456. Citado na página 30.

Jin, J.; Dundar, A.; Culurciello, E. Flattened Convolutional Neural Networks for Feedforward Acceleration. *arXiv e-prints*, p. arXiv:1412.5474, dez. 2014. Citado na página 27.

JUNEJO, I.; AHMED, N. Depthwise separable convolutional neural networks for pedestrian attribute recognition. *SN Computer Science*, v. 2, 04 2021. Citado 2 vezes nas páginas 10 e 28.

Kaiser, L.; Gomez, A. N.; Chollet, F. Depthwise Separable Convolutions for Neural Machine Translation. *arXiv e-prints*, p. arXiv:1706.03059, jun. 2017. Citado na página 28.

KANG, H.-J. Ssdlitex: Enhancing ssdlite for small object detection. *Applied Sciences*, v. 13, n. 21, 2023. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/13/21/12001>. Citado 2 vezes nas páginas 10 e 53.

KAUR, B.; SINGH, S. Object detection using deep learning: A review. In: *Proceedings of the International Conference on Data Science, Machine Learning and Artificial Intelligence*. New York, NY, USA: Association for Computing Machinery, 2022. (DSMLAI '21'), p. 328–334. ISBN 9781450387637. Disponível em: <https://doi.org/10.1145/3484824.3484889>. Citado na página 49.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. In: BENGIO, Y.; LECUN, Y. (Ed.). *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. [s.n.], 2015. Disponível em: <http://arxiv.org/abs/1412.6980>. Citado na página 33.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 60, n. 6, p. 84–90, may 2017. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/3065386>. Citado na página 19.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, v. 521, n. 7553, p. 436–444, 2015. Disponível em: <https://doi.org/10.1038/nature14539>. Citado 2 vezes nas páginas 16 e 19.

LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, v. 1, n. 4, p. 541–551, 1989. Citado na página 19.

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft coco: Common objects in context. In: FLEET, D.; PAJDLA, T.; SCHIELE, B.; TUYTELAARS, T. (Ed.). *Computer Vision – ECCV 2014*. Cham: Springer International Publishing, 2014. p. 740–755. ISBN 978-3-319-10602-1. Citado 2 vezes nas páginas 35 e 61.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.-Y.; BERG, A. C. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, Springer International Publishing, p. 21–37, 2016. ISSN 1611-3349. Disponível em: <http://dx.doi.org/10.1007/978-3-319-46448-0_2>. Citado 9 vezes nas páginas 10, 16, 17, 50, 53, 54, 55, 57 e 61.

LORENTE, Ò.; RIERA, I.; RANA, A. Image classification with classic and deep learning techniques. *CoRR*, abs/2105.04895, 2021. Disponível em: <https://arxiv.org/abs/2105.04895>. Citado na página 16.

LOWE, D. Object recognition from local scale-invariant features. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. [S.l.: s.n.], 1999. v. 2, p. 1150–1157 vol.2. Citado na página 49.

MAK, D. K. Exponential moving averagemoving average. In: _____. *Trading Tactics in the Financial Market: Mathematical Methods to Improve Performance*. Cham: Springer International Publishing, 2021. p. 57–71. ISBN 978-3-030-70622-7. Disponível em: <https://doi.org/10.1007/978-3-030-70622-7_4>. Citado na página 33.

MCCULLOCH, W.; PITTS, W. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, v. 5, p. 127–147, 1943. Citado na página 19.

MOURA, G.; SOUZA, K.; NETO, J. C.; TERNES, S. Comparando a eficiência de redes neurais convolutivas para detecção de frutos em pomares de laranja. In: *Anais do XIV Congresso Brasileiro de Agroinformática*. Porto Alegre, RS, Brasil: SBC, 2023. p. 414–419. ISSN 2177-9724. Disponível em: <https://sol.sbc.org.br/index.php/sbiagro/article/view/26587>. Citado na página 18.

NETO, J. C.; TERNES, S.; SOUZA, K. X. S. d.; YANO, I. H.; QUEIROS, L. R. Uso de redes neurais convolucionais para detecção de laranjas no campo. In: MARIA FERNANDA MOURA, JAYME GARCIA ARNAL BARBEDO, ALAINE MARGARETE GUIMARãES, VALTER CASTELHANO DE OLIVEIRA. *Anais do Congresso Brasileiro de Agroinformática*. Ponta Grossa: SBIAGRO, 2019. p. 312–321. ISBN 978-65-00-10242-0. Embrapa Agricultura Digital, 21/10/2020. Atualizado em 22/10/2020. Disponível em: <https://ainfo.cnptia.embrapa.br/digital/bitstream/item/216886/1/PC-Redes-neurais-SBIAGRO-2019.pdf>. Citado na página 17.

NG, A. Y. Feature selection, l1 vs. l2 regularization, and rotational invariance. In: *Proceedings of the Twenty-First International Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2004. (ICML '04), p. 78. ISBN 1581138385. Disponível em: <https://doi.org/10.1145/1015330.1015435>. Citado na página 34.

NIRTHIKA, R.; MANIVANNAN, S.; RAMANAN, A.; WANG, R. Pooling in convolutional neural networks for medical image analysis: A survey and an empirical study. *Neural Comput. Appl.*, Springer-Verlag, Berlin, Heidelberg, v. 34, n. 7, p. 5321–5347, apr 2022. ISSN 0941-0643. Disponível em: <https://doi.org/10.1007/s00521-022-06953-8>. Citado na página 27.

NOURA, H. N.; SALMAN, O.; COUTURIER, R.; SIDER, A. A deep learning object detection method for an efficient clusters initialization. *CoRR*, abs/2104.13634, 2021. Disponível em: <https://arxiv.org/abs/2104.13634>. Citado na página 17.

PADILLA, R.; PASSOS, W. L.; DIAS, T. L. B.; NETTO, S. L.; SILVA, E. A. B. da. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, v. 10, n. 3, 2021. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/10/3/279>. Citado 2 vezes nas páginas 36 e 41.

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)*. Los Alamitos, CA, USA: IEEE Computer Society, 2016. p. 779–788. ISSN 1063-6919. Disponível em: <https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.91>. Citado 3 vezes nas páginas 16, 50 e 51.

REDMON, J.; FARHADI, A. Yolo9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 6517–6525, 2016. Disponível em: <https://api.semanticscholar.org/CorpusID:786357>. Citado na página 17.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In: CORTES, C.; LAWRENCE, N.; LEE, D.; SUGIYAMA, M.; GARNETT, R. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015. v. 28. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf>. Citado 2 vezes nas páginas 51 e 52.

RIBEIRO, A. F.; SOUZA, K. X. S. d.; NETO, J. C.; TERNES, S.; YANO, I. H. Testando a rede neural yolov5 para detecção de frutos em pomares de laranja. In: . Campinas: CNPTIA, 2022. p. 22602. ISBN 978-65-88414-07-1. Disponível em: <http://www.alice.cnptia.embrapa.br/alice/handle/doc/1146123>. Citado na página 17.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, n. 6, p. 386–408, 1958. ISSN 0033-295X. Disponível em: <http://dx.doi.org/10.1037/h0042519>. Citado 2 vezes nas páginas 19 e 20.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group UK London, v. 323, n. 6088, p. 533–536, 1986. Citado na página 34.

RUMELHART, D. E.; MCCLELLAND, J. L.; GROUP, C. P. R. (Ed.). *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*. Cambridge, MA, USA: MIT Press, 1986. ISBN 026268053X. Citado na página 21.

SAKKA, J. M. M. E.; IVANOVICI, M. Images and cnn applications in smart agriculture. *European Journal of Remote Sensing*, Taylor & Francis, v. 57, n. 1, p. 2352386, 2024. Disponível em: <https://doi.org/10.1080/22797254.2024.2352386>. Citado na página 17.

SANDLER, M.; HOWARD, A.; ZHU, M.; ZHMOGINOV, A.; CHEN, L. Mobilenetv2: Inverted residuals and linear bottlenecks. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, 2018. p. 4510–4520. Disponível em: <https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00474>. Citado 6 vezes nas páginas 10, 16, 18, 29, 46 e 53.

SHAN, B.; FANG, Y. A cross entropy based deep neural network model for road extraction from satellite images. *Entropy*, v. 22, n. 5, 2020. ISSN 1099-4300. Disponível em: <https://www.mdpi.com/1099-4300/22/5/535>. Citado na página 32.

SHIRAZI, A. S.; FRIGAARD, I. Slurrynet: Predicting critical velocities and frictional pressure drops in oilfield suspension flows. *Energies*, v. 14, p. 1263, 02 2021. Citado 2 vezes nas páginas 10 e 22.

SHUKLA, A.; AANAND, A.; NITHIYA, S. Automatic speech recognition using machine learning techniques. In: *2023 International Conference on Computer Communication and Informatics (ICCCI)*. [S.l.: s.n.], 2023. p. 1–6. Citado na página 16.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. In: . [S.l.]: Computational and Biological Learning Society, 2015. p. 1–14. Citado 3 vezes nas páginas 18, 44 e 52.

SOUSA, M. A. d.; SOUZA, K. X. S. d.; NETO, J. C.; TERNES, S.; YANO, I. H. Usando a rede neural ssd para identificar frutos verdes em pomares de laranja. In: . Campinas: CNPTIA, 2021. p. 21610. ISBN 978-65-994972-0-9. Disponível em: <https://ainfo.cnptia.embrapa.br/digital/bitstream/item/226801/1/RE21610.pdf>. Citado na página 17.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 56, p. 1929–1958, 2014. Disponível em: <http://jmlr.org/papers/v15/srivastava14a.html>. Citado 3 vezes nas páginas 10, 34 e 35.

SZELISKI, R. *Computer vision algorithms and applications*. London; New York: Springer, 2011. Disponível em: <http://dx.doi.org/10.1007/978-1-84882-935-0>. Citado na página 16.

VIOLA, P.; JONES, M. J. Robust real-time face detection. *International Journal of Computer Vision*, Springer Science and Business Media LLC, v. 57, n. 2, p. 137–154, maio 2004. Disponível em: <https://doi.org/10.1023/b:visi.0000013087.49260.fb>. Citado na página 49.

WIDROW, B.; LEHR, M. 30 years of adaptive neural networks: Perceptron and madaline and and backpropagation. *Proceedings of the IEEE*, v. 78, n. 9, p. 1415–1442, 1990. Citado na página 20.