UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA

Classificação de Imagens de Frutas utilizando Aprendizado de Máquina

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Engenharia Eletrônica.

Ayrton Lima da Rosa

Orientador: Prof. Danilo Silva, Ph.D.

Florianópolis, Julho de 2019.

AYRTON LIMA DA ROSA

CLASSIFICAÇÃO DE IMAGENS DE FRUTAS UTILIZANDO APRENDIZADO DE MÁQUINA

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador: Prof. Danilo Silva, Ph.D.

FLORIANÓPOLIS
2019

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Rosa, Ayrton
Classificação de Imagens de Frutas utilizando Aprendizado de Máquina / Ayrton Rosa ; orientador, Danilo Silva , 2019.
74 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Eletrônica, Florianópolis, 2019.

Inclui referências.

1. Engenharia Eletrônica. 2. Aprendizado de Máquina. 3. Classificação de Frutas. 4. Redes Neurais Convolucionais. I. , Danilo Silva. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Eletrônica. III. Título.

Ayrton Lima da Rosa

CLASSIFICAÇÃO DE IMAGENS DE FRUTAS UTILIZANDO APRENDIZADO DE MÁQUINA

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do Título de Bacharel em Engenharia Eletrônica, e aprovado em sua forma final pelo Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina.

Florianópolis, 12 de Julho de 2019.

Prof. Jefferson Luiz Brum Marques, Ph.D.
Coordenador do Curso
Universidade Federal de Santa Catarina

Banca examinadora:

Prof. Danilo Silva, Ph.D.
Orientador
Universidade Federal de Santa Catarina

Prof. Joceli Mayer, Ph.D.
Universidade Federal de Santa Catarina

Prof. Marcio Holsbach Costa, Dr.
Universidade Federal de Santa Catarina

Agradecimentos

Desejo expressar meu reconhecimento a todos que, de uma maneira ou outra, colaboraram na realização deste trabalho. Agradeço à minha família por todo apoio que tive durante a graduação. Ao meu pai Marco Antônio e minha mãe Cleonice, pelo amor e apoio incondicional. Ao meu irmão Guilherme, pelo incentivo nas horas difíceis. À minha namorada Vanessa, sempre me apoiando nos momentos bons e ruins, mesmo estando longe. Aos meus amigos que conheci na universidade, por todos esses anos de amizade e parceria.

Aos meus professores, por proporcionarem uma educação de qualidade. Em especial, um agradecimento ao professor Danilo Silva, pela orientação, apoio e confiança durante esse trabalho.

Por fim, agradeço à todos que de alguma forma contribuíram para minha formação profissional.

RESUMO

Este trabalho tem como propósito utilizar técnicas de aprendizado de máquina para classificação de imagens de frutas. O objetivo deste trabalho é atingir uma acurácia superior a 90% no conjunto que será testado. Foram utilizados modelos de redes neurais convolucionais utilizando a biblioteca *Keras* em *Python*.

Os modelos de aprendizado de máquina escolhidos para este trabalho são modelos pré-treinados disponíveis na biblioteca *Keras*. Esses modelos foram treinados em um grande conjunto de imagens chamado *ImageNet*, e serão aplicados nesse trabalho utilizando o método de transferência de aprendizagem. Além disso, também foi construído um modelo utilizando redes neurais convolucionais a fim de comparação com os resultados encontrados com os outros modelos.

O conjunto de dados inicialmente escolhido foi o *Fruit Dataset*, composto por 20 classes de objetos. As imagens do conjunto de dados foram usadas diretamente como entrada da rede neural para treinamento e aplicação. Os resultados obtidos mostram que a abordagem proposta valida a utilização de redes neurais para classificação de frutas com uma alta acurácia.

Palavras-chave: Aprendizado de Máquina, Redes Neurais Convolucionais, Classificação de Frutas.

ABSTRACT

This work aims to use machine learning techniques to classify fruit images. The objective of this work is to reach an accuracy rate higher than 90% in the dataset that will be tested. Convolutional neural network models were used using the *Keras* library in *Python*.

The machine learning models selected for this work are pre-trained models available in the *Keras* library. These models were trained in a large dataset of images called *ImageNet*, and will be applied in this work using the transfer learning method. In addition, a model was also constructed using convolutional neural networks in order to compare with the results found with the other models.

The dataset initially chosen was the *Fruit Dataset*, composed of 20 classes of objects. The images were used directly as neural network input for training and application. The results show that the proposed approach validates the use of neural networks for fruit classification with high accuracy.

Keywords: Machine Learning, Convolutional Neural Networks, Fruit Classification.

Lista de Figuras

2.1 Sistema RGB e o conceito de *pixels*. 6
2.2 Sistema HSV e suas componentes. 7
2.3 Tarefas realizadas na área de reconhecimento da visão computacional. 8
2.4 Configuração da rede neural simples e da rede neural profunda. 9
2.5 Estrutura de uma Rede Neural Convolucional. 10
2.6 Exemplo de *overfitting*. 12

3.1 Diagrama de blocos do processo de classificação. 16
3.2 Exemplos de imagens do *dataset*. 18
3.3 Resultados apresentados em [1], utilizando combinação de características. C e S representam cor e textura, respectivamente. 19
3.4 Resultados apresentados em [2], utilizando combinação de características de cor e textura. 19
3.5 Estrutura da rede convolucional. 21
3.6 Resultados apresentados em [3], utilizando diferentes cenários de pré-processamento das imagens. 22
3.7 Semelhança entre o conjunto de treinamento e de teste no *dataset Fruits-360*. 22

4.1 Exemplos de *data augmentation*, a imagem original se encontra no topo. 30
4.2 Estrutura dos modelos VGG16 e VGG19. 31
4.3 Módulo Inception. 32
4.4 Estrutura da Rede Inception. 33
4.5 Modelo de convoluções pontuais seguidas de convoluções em profundidade. 33
4.6 Arquitetura básica de uma rede convolucional. 35
4.7 Exemplo de predições. 37

5.1 Curvas da acurácia e da perda durante o treinamento do modelo VGG16 com o *dataset original*. 42
5.2 Curvas da acurácia e da perda durante o treinamento do modelo VGG16 com o *dataset original* e com a melhor configuração. 46

Lista de Tabelas

3.1 *Dataset* utilizado em [1]. 17

4.1 *Dataset original* separado em conjuntos de treinamento, validação e teste. 27
4.2 *Dataset* utilizado em [1] ampliado. 28
4.3 Redes neurais e suas respectivas acurácias no *dataset ImageNet*. 31

5.1 Resultado da rede convolucional com o *dataset Fruits-360*. 40
5.2 Resultados da rede convolucional com o *dataset original* e com o *dataset ampliado*. 40
5.3 Resultados da rede convolucional com o *dataset original* e com o *dataset ampliado* utilizando *transfer learning*. 41
5.4 Resultados dos modelos pré-treinados com o *dataset original* utilizando *transfer learning*. 42
5.5 Resultados dos modelos pré-treinados com o *dataset ampliado* utilizando *transfer learning*. 42
5.6 Resultados do modelo VGG16 utilizando diferentes otimizadores, com o conjunto de validação do *dataset original*. . 43

5.7 Resultados com o otimizador *RMSprop* no conjunto de validação utilizando diferentes métodos de redução da taxa de aprendizado. 44
5.8 Resultados do modelo VGG16 no conjunto de validação utilizando dois classificadores e dois métodos de *transfer learning*. 45
5.9 Resultados utilizando o classificador 2 e o método *transfer learning* 2 nos conjuntos de teste do *dataset original* e do *dataset ampliado*. 46
5.10 Resultados ao treinar toda a rede VGG16 com os *datasets original e ampliado*. 46

Lista de Abreviaturas

Abreviatura	Descrição
Concat	Camada de concatenação.
AvgPool	Camada de agrupamento que realiza a operação de agrupamento médio.
MaxPool	Camada de agrupamento que realiza a operação de agrupamento máximo.
Conv3	Camada de convolução de três dimensões que realiza convolução espacial sobre volumes. Possui também um número inteiro associado, que representa a dimensionalidade do espaço de saída.
FC	Camada Totalmente Conectada. Possui também um número inteiro associado, que representa a dimensionalidade do espaço de saída.
Softmax	Função de ativação utilizada principalmente em aplicações multi classe.

Sumário

1 Introdução **1**

1.1 Objetivos Gerais 2
1.2 Objetivos Específicos 3
1.3 Estrutura do Trabalho 3

2 Fundamentação Teórica **5**

2.1 Visão Computacional 5
2.2 Aprendizado de Máquina 8
2.2.1 Redes Neurais Artificiais 9
2.2.2 Redes Neurais Convolucionais 10
2.2.3 Treinamento 11
2.2.4 *Overfitting* 11
2.2.5 Modelos pré-treinados 12

3 Métodos Existentes na Literatura **15**

3.1 Reconhecimento usando métodos clássicos 15
3.2 Reconhecimento usando Aprendizado de Máquina 20

4 Método Proposto **25**

4.1 *Dataset* 25
4.1.1 Obtenção das imagens 26
4.1.2 Balanceamento do *dataset* 28

4.1.3 *Data Augmentation* 29
4.2 Escolha das Redes Neurais 30
4.2.1 VGG 31
4.2.2 InceptionV3 32
4.2.3 Xception 32
4.2.4 MobileNetV2 33
4.3 Configuração das Redes Neurais 34
4.3.1 Transferência de aprendizagem 34
4.4 Método de avaliação de desempenho 36
4.5 *Software* de Processamento 37

5 Resultados **39**
5.1 Resultados dos modelos escolhidos 39
5.1.1 Rede Neural Convolucional 40
5.1.2 Modelos pré-treinados 41
5.2 Simulações com o melhor método 42
5.3 Discussão dos resultados 47

6 Conclusão **49**
6.1 Trabalhos futuros 50

Referências bibliográficas **51**

CAPÍTULO 1

Introdução

O desenvolvimento de novas tecnologias vem crescendo rapidamente a cada ano. Todo dia surgem novas oportunidades de aumentar a produtividade do trabalho e na gestão inteligente de recursos. Nos últimos anos, um dos setores que obteve grandes vantagens com inovações tecnológicas foi o setor de serviços, principalmente com novas tecnologias de análise, reconhecimento e identificação de produtos.

Um exemplo disso é o setor de supermercados, considerado como setor tradicional, que está se modernizando em busca de uma melhor experiência ao usuário. Nessa modernização, encontra-se o uso da tecnologia de *selfcheck-out*, ou caixas de autoatendimento, que já está sendo implementada em várias redes de supermercados no Brasil. Essa tecnologia permite que o próprio consumidor realize o processo de registrar as suas mercadorias e realizar o pagamento desses itens. Porém a grande maioria dos supermercados ainda utilizam caixas tradicionais, que requerem que o operador memorize códigos de produtos que não possuem códigos de barra para sua identificação, principalmente produtos de hortifrúti. Em adição a esse problema, é importante ressaltar que a qualidade e rapidez de atendimento formam as características mais apontadas pelos clientes como motivo de frequência ao supermer-

cado [4]. É importante notar que estas duas características sao mais importantes que os preços dos produtos, na opinião dos clientes.

Uma das consequências de toda essa inovação e modernização, foi o aumento no volume de dados, tornando-se necessário o uso de uma inteligência de gestão capaz de processar esses dados e transformá-los em informações úteis e lucrativas. Um exemplo disso é o caso do *Amazon Go* [5], um novo tipo de loja onde não há a necessidade de *check-out*, isto é, o consumidor não precisa de nenhuma interação com os funcionários da loja ou com máquinas, para que a compra seja efetuada. Nele são usadas tecnologias avançadas, como visão computacional, fusão de sensores e aprendizado de máquina profundo. Para realizar o processamento de grandes volumes de dados existentes atualmente, existem diversas formas, e umas das áreas que mais tem avançado nesse ramo é a de aprendizado de máquina.

Os avanços nessa área, principalmente devido à criação de conjuntos de dados diversos, novos algoritmos de treinamento de redes neurais profundas e *hardware* mais potente, possibilitaram o desenvolvimento de algoritmos extremamente inteligentes, capazes de realizar tarefas de alto nível cognitivo.

Com o objetivo de combinar essas inovações tecnológicas e incentivar o desenvolvimento tecnológico do país, este trabalho é focado na utilização de redes neurais complexas para a classificação de imagens de frutas, possibilitando uma inovação para ser utilizada em redes de supermercados tradicionais, que irá ajudar na detecção em tempo real de produtos, aumentando a rapidez do atendimento, sem que para isso necessite de intervenção humana.

1.1 Objetivos Gerais

O presente trabalho tem como objetivo a avaliação e aplicação de modelos modernos de aprendizado de máquina que utilizam redes neurais convolucionais para classificar 20 categorias de objetos, obtendo uma acurácia maior que 90% no conjunto de teste. A fim de comparação, será utilizado o conjunto de dados presente em [1], com algumas modificações, em busca de melhores resultados.

1.2 Objetivos Específicos

- Analisar, dividir e enriquecer o *dataset* proposto em [1], em busca de um melhor desempenho utilizando redes neurais convolucionais.
- Escolher, treinar e avaliar diferentes modelos de redes neurais convolucionais para classificação de imagens de frutas, e comparar com métodos existentes na literatura.
- Analisar a viabilidade de utilização dos modelos estudados para o uso cotidiano e comercial.

1.3 Estrutura do Trabalho

Este trabalho está dividido em quatro partes, iniciando por uma revisão teórica dos conceitos utilizados, de forma a analisar o problema, verificar a viabilidade de execução e fundamentar o desenvolvimento do trabalho. Na segunda parte, são explicadas as metodologias já existentes que lidam com o problema utilizando aprendizado de máquina e também utilizando o modelo clássico que utiliza visão computacional.

Em seguida, são explicadas as metodologias de desenvolvimento do trabalho, considerando todos os aspectos, desde a definição do conjunto de dados até as metodologias de avaliação utilizadas. Após isso, são analisados os resultados obtidos e comparados com os métodos existentes, de forma a chegar a uma conclusão sobre os métodos utilizados neste trabalho. Por fim, são realizadas conclusões sobre os resultados obtidos, e são discutidas possíveis melhorias na metodologia utilizada e trabalhos futuros que poderão ser realizados sobre o tema.

CAPÍTULO 2

Fundamentação Teórica

Neste trabalho foram utilizados conhecimentos da área de aprendizado de máquina, principalmente com ênfase em redes neurais convolucionais. Nas seções seguintes, serão descritos todos os conhecimentos utilizados nesse trabalho, explicando os conceitos e as diferenças existentes entre as redes neurais utilizadas. Além disso, também serão apresentados alguns conceitos de visão computacional, para uma melhor compreensão quando os modelos desse trabalho forem comparados com métodos clássicos de classificação de objetos em imagens.

2.1 Visão Computacional

A visão computacional é um campo multidisciplinar que tem como objetivo usar máquinas para entender e analisar imagens (fotos e vídeos, por exemplo). A base teórica dessa área já vem sendo elaborada há bastante tempo, porém com os avanços tecnológicos na área de computação é possível obter importantes melhorias no desempenho de métodos de processamento de imagens. Alguns exemplos de aplicações incluem o controle de processos industriais, detecção de objetos e modelagem de objetos ou ambientes.

Mesmo sendo seu conceito fácil de entender, a tarefa da visão computacional de "ensinar" uma máquina a analisar imagens, se tornou uma tarefa constantemente desafiadora. Uma razão para isso é que o processo da visão humana ainda não está totalmente compreendido. Uma outra razão é a complexidade presente no mundo visual. Um determinado objeto pode ter muitas peculiaridades quando representado em uma imagem, como por exemplo: várias orientações, diferentes condições de iluminação e interferência de outros objetos. Os métodos atuais funcionam de forma adequada para problemas específicos, mas ainda estão longe de serem utilizáveis para qualquer tipo de aplicação.

Para a máquina, as imagens são conjuntos de pequenos espaços, chamados *pixels*. Cada pixel pode ser representado por um número, geralmente entre 0 a 255, que indica a sua intensidade. Para imagens coloridas, cada pixel pode ser representado por 3 valores (vermelho, verde e azul) na mesma escala de 0 a 255. Esse sistema de cores se chama RGB (do inglês: *red*, *green* e *blue*), onde as três cores são combinadas de várias formas de modo a reproduzir um largo espectro cromático. O formato RGB pode ser visto na Figura 2.1, juntamente com o conceito de *pixels*.

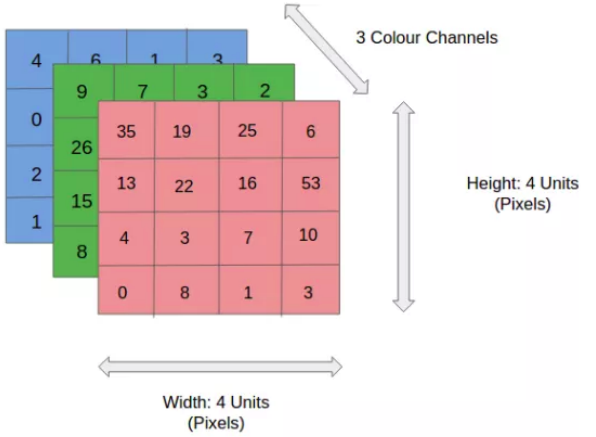

Figura 2.1: Sistema RGB e o conceito de *pixels*. Fonte: [6].

Um outro sistema para representar imagens é o HSV (do inglês: *hue* (matiz), *saturation* (saturação) e *value* (brilho)). As componentes

do sistema HSV podem ser vistas na Figura 2.2. A representação em HSV é frequentemente selecionada por suas propriedades invariantes. Por exemplo, o uso da componente matiz pode tornar o método menos sensível às variações de iluminação. Assim, cada representação pode ser útil em cada tipo de aplicação.

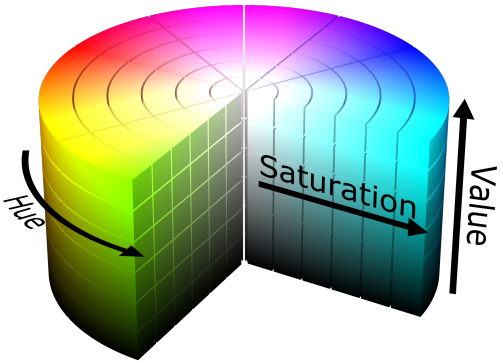

Figura 2.2: Sistema HSV e suas componentes. Fonte: [7].

Após a correta leitura da imagem, podem ser realizadas diversas transformações de visão computacional, para extrair informações das características presentes na imagem. Esse processo de extração envolve simplificar os dados de entrada ao descrever a imagem usando suas características, como por exemplo, usar métodos para detectar partes ou formas relevantes em uma imagem e com isso diminuir a quantidade de processamento.

Dentro das áreas que abrangem a visão computacional, a de reconhecimento é a que será abordada nesse trabalho. Ela consiste na detecção, classificação e identificação de objetos em uma imagem. A classificação de imagens pode ser definida como a tarefa de categorizar imagens em uma das várias classes previamente escolhidas. Ela forma a base para outras tarefas de visão computacional, como localização, detecção e segmentação [8], como mostra a Figura 2.3. Atualmente, os métodos de classificação mais eficientes são as redes neurais convolucionais, um subcampo da área de aprendizado de máquina.

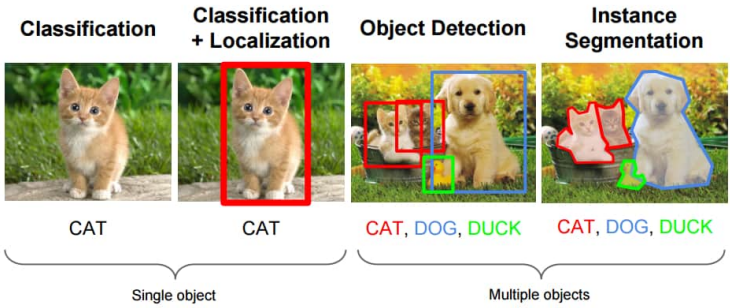

Figura 2.3: Tarefas realizadas na área de reconhecimento da visão computacional. Fonte: [9].

2.2 Aprendizado de Máquina

O aprendizado de máquina é o uso de métodos computacionais para coletar dados, aprender padrões com eles, e então fazer uma determinação ou predição utilizando esses padrões. Isto é, a máquina consegue ser "treinada" usando uma quantidade de dados e métodos que dão a ela habilidades de aprender como executar a tarefa, sem ser explicitamente programada para isso [10]. As tarefas de treinamento de aprendizado de máquina podem ser divididas em três campos, de acordo com o tipo de *feedback* utilizado no treinamento. São eles:

(*i*) **Aprendizado supervisionado**: são fornecidas entradas e saídas exemplos para o computador através do conjunto de dados, com o objetivo de ensiná-lo a mapear uma regra geral. Esta é a metodologia utilizada neste trabalho.

(*ii*) **Aprendizado não supervisionado**: não são fornecidas saídas mapeadas e o objetivo da rede é identificar padrões, para que a parte de aprendizado seja feita pela própria máquina.

(*iii*) **Aprendizado por reforço**: a máquina interage com o ambiente de forma dinâmica e recebe um *feedback* baseado em uma função de avaliação para ajustar seus pesos e melhorar a cada iteração.

2.2.1 Redes Neurais Artificiais

Uma área importante do aprendizado de máquina são as redes neurais artificiais. Elas são sistemas inspirados no cérebro humano e em seus neurônios, que são capazes de realizar o aprendizado de máquina bem como o reconhecimento de padrões de forma eficiente ao adquirir conhecimento através da experiência.

As redes neurais possuem unidades de processamento, e cada uma dessas unidades, possui ligações para outras unidades, nas quais recebem e enviam sinais. Basicamente, é uma simulação dos neurônios presentes no cérebro, recebendo e retransmitindo informações.

A construção das redes neurais consiste basicamente de camadas de entrada e saída, e no mínimo uma camada intermediária. Geralmente, quanto maior o número de camadas da rede, melhor será a capacidade de aprendizado. Isto é, um número grande de camadas pode melhorar a capacidade de representação das relações entre o espaço de entrada e o de saída. Com poucas camadas intermediárias, o algoritmo pode não conseguir representar corretamente as relações não-lineares. A existência de múltiplas camadas intermediárias, amplia o universo de representação do método a qualquer função. Essas redes que possuem múltiplas camadas intermediárias são chamadas redes neurais profundas (do inglês: *Deep Learning*), como mostrado na Figura 2.4. Elas são excelentes ferramentas para encontrar padrões que são muito complexos ou numerosos para um programador humano extrair e ensinar a máquina a reconhecer.

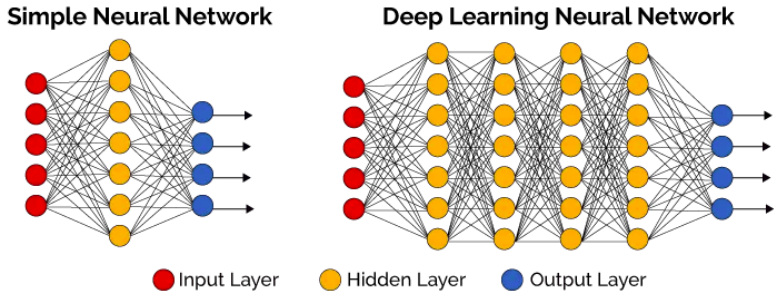

Figura 2.4: Configuração da rede neural simples e da rede neural profunda. Fonte: [11].

2.2.2 Redes Neurais Convolucionais

Na área de reconhecimento e classificação de imagens, os melhores resultados são obtidos usando as redes neurais convolucionais (CNNs, do inglês *Convolutional Neural Network*), que são uma subclasse das redes neurais profundas. Uma rede neural convolucional difere de uma rede neural regular pelo uso de no mínimo uma camada convolucional. Uma rede convolucional reconhece um objeto procurando primeiro por características de baixo nível, como bordas, linhas e curvas, e então segue construindo características mais abstratas através de uma série de camadas convolucionais [12]. Assim, essas redes conseguem reconhecer padrões extremamente complexos e são adaptáveis a distorções e variações das imagens. A Figura 2.5 ilustra a estrutura de uma CNN.

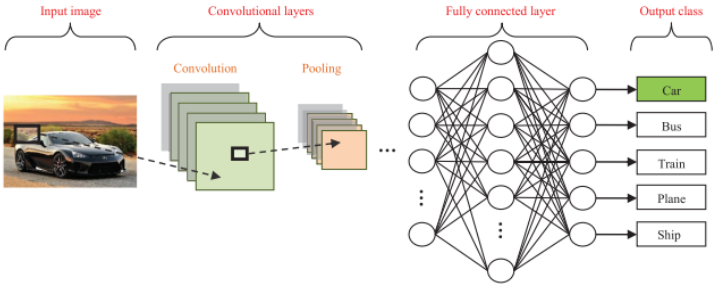

Figura 2.5: Estrutura de uma Rede Neural Convolucional. Fonte: [8].

As CNNs são redes do tipo *feed-forward*, em que o fluxo de informações ocorre em apenas uma direção, desde suas entradas até suas saídas [8]. Uma simples rede convolucional é uma sequência de camadas, e cada camada transforma um volume de ativações em outra camada através de uma função diferenciável. Basicamente, usam-se três tipos principais de camadas para construir uma CNN: camada convolucional, camada de agrupamento (do inglês: *Pooling layer*) e camada totalmente conectada.

As camadas convolucionais servem como extratores de características e, assim, aprendem as representações características de suas imagens de entrada. Os neurônios nessas camadas são organizados em mapas de características (do inglês: *feature maps*). Logo após as camadas convolucionais, existem as camadas de *pooling*, cuja finalidade

é reduzir a resolução espacial das *feature maps* para obter invariância espacial sobre distorções e deslocamentos.

Uma imagem é inserida diretamente na rede e isso é seguido por várias camadas de convolução e agrupamento. Posteriormente, as representações dessas camadas alimentam uma ou mais camadas totalmente conectadas (do inglês: *fully connected layers*). As camadas totalmente conectadas interpretam essas representações e executam o processamento de alto nível. Finalmente, a última camada totalmente conectada produz o rótulo da classe do objeto [8]. Todos os parâmetros estruturais da CNN, como por exemplo: tamanho das imagens de entrada, número e tamanho de camadas, tamanho dos filtros, entre outros, são adaptáveis para qualquer aplicação particular [13].

2.2.3 Treinamento

Uma maneira de avaliar o desempenho de uma rede é através da avaliação da chamada função custo. O processo de treinamento de uma rede neural é justamente o problema de otimização dessa função. A função de perda é um método de avaliação de como o método modela os dados fornecidos, representando o "preço pago" pelos erros das previsões do modelo. Assim, com a ajuda de algum otimizador, a função de perda serve como um guia para reduzir o erro na previsão.

Neste trabalho foi utilizada a função de perda de entropia cruzada (do inglês: *categorical cross-entropy*) presente na biblioteca *Keras*. É a função de perda mais comum para problemas de classificação. A perda aumenta à medida que a probabilidade prevista diverge da classe real, e penaliza fortemente as previsões que se mostram confiantes, mas erradas.

2.2.4 *Overfitting*

Durante o treinamento de modelos de aprendizado de máquina, é comum ocorrer um fenômeno chamado sobreajuste (do inglês: *overfitting*). Isso ocorre quando o modelo treinado se ajusta muito bem ao conjunto de dados de treinamento, porém é ineficaz ao tentar classificar novos dados. Um modelo com *overfitting* apresenta alta acurácia quando testado em seu próprio conjunto de dados, porém esse modelo não é uma representação confiável da realidade. Na Figura 2.6 pode-se

observar um modelo de predição em que a linha verde representa um modelo com *overfitting* e a linha preta um modelo bem ajustado.

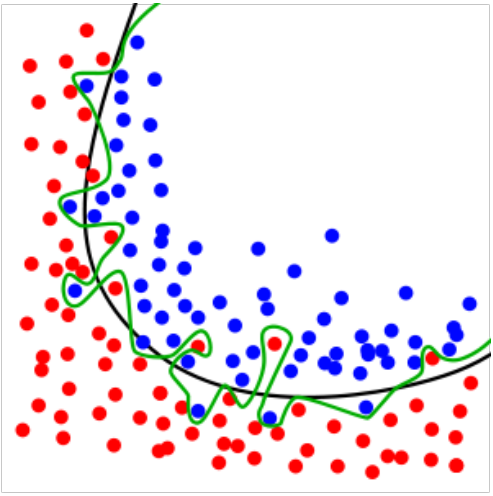

Figura 2.6: Exemplo de *overfitting*. Fonte: [14].

Uma forma de evitar o *overfitting* e que foi usada neste trabalho, é a parada antecipada (do inglês: *early stopping*). Ela é uma forma de regularização usada durante o treinamento, isto é, o modelo é avaliado em um conjunto de dados de validação após cada época de treinamento. Se o desempenho do modelo nesse conjunto começar a degradar, o processo de treinamento é interrompido.

2.2.5 Modelos pré-treinados

Desde o surgimento das redes neurais profundas, principalmente utilizando redes convolucionais, o Desafio de Reconhecimento Visual de Grande Escala do *ImageNet* (ILSVRC, do inglês: *ImageNet Large Scale Visual Recognition Challenge*) desempenhou um papel importante no avanço do estado da arte [15]. O ILSVRC é uma competição anual organizada pela equipe ImageNet desde 2010, onde as equipes de pesquisa avaliam seus algoritmos de visão computacional e aprendizado de máquina em várias tarefas de reconhecimento visual, como classificação e localização de objetos. Os dados de treinamento são um subconjunto do conjunto de dados *ImageNet* com 1,2 milhão de imagens perten-

centes a 1000 classes. O *ImageNet* é um projeto que visa fornecer um grande banco de imagens para fins de pesquisa, e contém mais de 14 milhões de imagens que pertencem a mais de 20.000 classes.

Ao se utilizar grandes bancos de dados, como o *ImageNet*, é possível treinar redes neurais profundas mais complexas, que geram excelentes resultados. Porém, para realizar todo esse processamento, é necessário um maquinário potente e mesmo assim, ainda gasta-se muito tempo para treinar essas redes. Felizmente, existem desafios como o ILSVRC, onde muitos grupos de pesquisa compartilham os modelos que foram treinados em um grande conjunto de dados, por um grande período e utilizando *hardwares* poderosos. Assim, pode-se basear nesses modelos pré-treinados, como ponto de partida para o processo de treinamento.

Para esse trabalho, serão usados 4 tipos de modelos que possuem código aberto para a comunidade e que já foram testados em competições como o ILSVRC. Eles são: VGGNet, MobileNet, Xception e Inception. A escolha desses modelos e suas características serão explicadas nas Seções 4.2 e 4.3.

CAPÍTULO 3

Métodos Existentes na Literatura

Este capítulo revisa os principais métodos utilizados para a classificação de imagens de frutas presentes na literatura. No ramo de classificação de objetos, pode-se, em geral separar os métodos entre dois tipos. O primeiro é uma abordagem tradicional, onde se utiliza visão computacional e classificadores simples. Já o segundo, é uma abordagem que se utiliza do aprendizado de máquina, em especial, as redes neurais convolucionais, para realizar a classificação. A seguir, são explicados como esses métodos funcionam e os trabalhos relacionados encontrados.

3.1 Reconhecimento usando métodos clássicos

A abordagem tradicional, em geral, pode ser descrita utilizando os processos mostrados na Figura 3.1. Primeiramente é realizada a aquisição das imagens que contenham os objetos a serem classificados. Em seguida, essas imagens são pré-processadas e então divididas em um conjunto de treinamento e um conjunto de teste. Após isso, é realizada a extração de características de cada imagem do conjunto de treinamento e um classificador é treinado utilizando esses resultados. Por fim, utiliza-se a mesma técnica para extrair as características de cada

imagem do conjunto de teste, para que seja corretamente classificada utilizando o classificador já treinado.

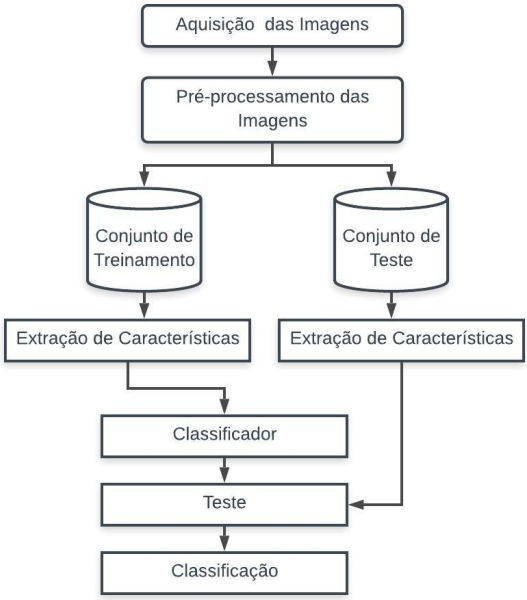

Figura 3.1: Diagrama de blocos do processo de classificação. Fonte: Adaptado de [2].

Os trabalhos encontrados na literatura, que se utilizam desse modelo tradicional de classificação, se diferem nos quesitos de escolha dos métodos de pré-processamento, dos métodos de extração das características e do classificador utilizado. A maioria dos trabalhos utiliza uma combinação das características extraídas para melhorar a acurácia da classificação, tendo em vista o desempenho limitado ao usar uma única característica em imagens complexas.

Em [1], é proposto um método de classificação baseado na combinação de características que é capaz de reconhecer 20 classes de objetos em um conjunto de dados personalizado (*Fruit Dataset* [16]) que possui uma variedade de fundos complexos. O método se baseia em encontrar a melhor combinação de características para realizar a classificação. Antes de realizar a combinação é feita uma busca dos melhores

parâmetros para cada característica, encontrando assim, uma configuração "ótima" para a extração de cada característica . Uma vez que as características "ótimas" tenham sido selecionadas, é realizada a fusão dessas características usando o método de pontuação ponderada, com base nos pesos aprendidos de cada característica para cada classe individual. Isso é feito, pois para cada classe, diferentes características demonstraram diferentes acurácias na classificação.

Classe do objeto	Treinamento	Teste
Background	740	50
Morango	450	50
Pêssego	434	50
Laranja	432	50
Maça Vermelha	418	50
Tomate	411	50
Banana	394	50
Kiwi	311	50
Pera	268	50
Romã	266	50
Melancia	239	50
Abacaxi	225	50
Uva	164	50
Carambola	161	50
Limão	149	50
Manga	149	50
Jaca	120	50
Pomelo	118	50
Melão	102	50
Mamão	59	50
Total	5610	1000

Tabela 3.1: *Fruit Dataset* utilizado em [1]. Fonte: Adaptado de [1].

O *Fruit dataset* utilizado em [1] consiste em imagens, retiradas do *Google Imagens*, contendo um único tipo de classe, com diferentes planos de fundo, e pode ser encontrado em [16]. São 20 tipos de classes

como mostra a Tabela 3.1. O conjunto de teste foi formado escolhendo aleatoriamente 50 imagens de cada tipo de classe. O conjunto de treinamento está desbalanceado, isto é, algumas classes possuem poucos exemplares comparadas com outras. Mesmo o conjunto de dados tendo uma grande diversidade e com imagens de fundo complexas, um conjunto de treinamento com poucos exemplares pode não conseguir demonstrar com clareza todos os possíveis cenários que o objeto possa a vir se encontrar. As imagens possuem tamanho 100×100 pixels e estão no formato RGB. A Figura 3.2 mostra alguns exemplos de imagens do *Fruit Dataset*.

Figura 3.2: Exemplos de imagens do *dataset*. Fonte: [16].

Em [1], as características extraídas de cada imagem foram Cor, Forma, Histograma de Gradientes Orientados (HOG, do inglês: *Histogram of oriented gradients*), Padrões Binários Locais (LBP, do inglês: *Local binary pattern*) e edgeLBP. Na Figura 3.3, podem ser encontrados os resultados utilizando cada uma dessas características sozinhas ou a combinação delas. O melhor resultado encontrado foi de 90.7% de acurácia quando foram usadas todas as características combinadas.

Em [2], para o pré-processamento das imagens é utilizado um eficiente método de segmentação para remoção do plano de fundo da imagem, chamado *GrabCut*. Na parte de extração, as características de cor e textura são extraídas de cada imagem, e a combinação dessas características será a entrada de um classificador que utiliza *SVM*. Utilizando um conjunto de 8 diferentes classes de frutas, esse método conseguiu uma acurácia de 83.33% de acertos no conjunto de teste, como pode ser visto na Figura 3.4. Como descrito em [2], o conjunto

Features	Accuracy (%)	Features	Accuracy (%)
C	55.6	C + S + LBP + HOG + edgeLBP	**90.7**
S	25.7	S + HOG	82.2
LBP	83.7	LBP + edgeLBP	86.8
HOG	78.6	C + S + HOG	84.4
EdgeLBP	77.8	C + S + LBP + HOG	88.7

Figura 3.3: Resultados apresentados em [1], utilizando combinação de características. C e S representam cor e textura, respectivamente. Fonte: [1].

de dados possui 30 imagens para cada classe de fruta, e foi dividido em 60% para treinamento e 40% para validação. Esse conjunto não está desbalanceado como o anterior, porém possui poucos exemplares de cada classe.

Extracted Features	**Description**	**Feature Dimension**	**Accuracy (%)**
Texture	Statistical features extracted from gray-level co-occurrence matrix	16	32.29
Color	Mean, standard deviation, skewness, and kurtosis	12	69.79
Combined	Texture & Color	28	83.33

Figura 3.4: Resultados apresentados em [2], utilizando combinação de características de cor e textura. Fonte: [2].

Em [17], o método utilizado também é a combinação das características de cor e textura. Para o pré-processamento, cada imagem é representada no formato *HSV*, onde os recursos de textura são calculados a partir da luminância do canal V, enquanto que os recursos de cores são calculados a partir dos canais de crominância H e S. Assim são obtidas 13 características de cada imagem que possibilitam a construção de um banco de dados para posterior classificação das imagens de teste, utilizando um classificador de miníma distância [18]. Nesse artigo, para a combinação das características, foi utilizada a simples concatenação de uma característica após a outra para obter um vetor longo de características. É um método tradicional e simples, mas eficaz, que conseguiu uma acurácia média de 86% de acertos no conjunto

de teste.

O conjunto de dados de [17] compreende 15 categorias diferentes de frutas, totalizando 2633 imagens. Essas imagens são divididas em conjunto de treinamento e testes na proporção 1:1. As imagens foram reunidas em vários momentos do dia e em dias diferentes para a mesma categoria, para que aumentasse a variabilidade e representasse um cenário mais realista. Além disso, as imagens possuem diferenças na angulação dos objetos e no número de elementos dentro da imagem.

A partir desses trabalhos encontrados, pode-se perceber que o reconhecimento de objetos utilizando o método tradicional de visão computacional, pode ser bastante eficaz dependendo do conjunto de dados escolhido e dos métodos de extração de características aplicados.

3.2 Reconhecimento usando Aprendizado de Máquina

A utilização de aprendizado de máquina para realizar reconhecimento e classificação de objetos em imagens se tornou algo bastante comum, principalmente quando se usa redes neurais convolucionais. As camadas convolucionais da CNN agem como o extratores de características explicados anteriormente, isto é, em uma CNN as imagens são diretamente utilizadas como entrada da rede para treinamento e, além disso, a CNN pode aprender as melhores características das imagens através do processo de adaptação.

Em [19], é proposto um sistema de classificação de frutas usando um modelo de rede neural convolucional para extrair características da imagem e implementar a classificação de frutas em um projeto específico. Além disso, o conjunto de dados criado para esse projeto é composto por imagens de frutas de 15 categorias, e as imagens são específicas para esse único projeto. A rede neural convolucional consiste em 3 camadas de convolução e agrupamento máximo (do inglês: *max pooling*) para melhorar a capacidade do sistema. E, por fim, utilizam-se camadas totalmente conectadas para realizar a classificação.

O método proposto reconheceu facilmente imagens de frutas com muitos desafios presentes no mundo real. O resultado alcançado foi de uma acurácia de 99%. Porém, é importante ressaltar que o conjunto de teste se assemelha bastante ao conjunto de treinamento, dado que o projeto é para uma aplicação específica.

Em [3], é apresentado um novo conjunto de dados de imagens de alta qualidade contendo frutas, chamado *Fruits-360*. Ele possui 65429 imagens de frutas divididos em 95 classes. As imagens foram obtidas filmando as frutas enquanto elas são girados por um motor e, em seguida, extraindo os quadros do vídeo. A partir desse *dataset*, uma rede neural convolucional é utilizada para realizar o processo de classificação das frutas. A estrutura da rede pode ser vista na Figura 3.5. A coluna *Dimensions* representa o tamanho do filtro e a dimensionalidade dos dados de entrada. A coluna *Output* representa a dimensionalidade do espaço de saída.

Layer type	Dimensions	Output
Convolutional	5 x 5 x 4	16
Max pooling	2 x 2 — Stride: 2	-
Convolutional	5 x 5 x 16	32
Max pooling	2 x 2 — Stride: 2	-
Convolutional	5 x 5 x 32	64
Max pooling	2 x 2 — Stride: 2	-
Convolutional	5 x 5 x 64	128
Max pooling	2 x 2 — Stride: 2	-
Fully connected	5 x 5 x 128	1024
Fully connected	1024	256
Softmax	256	60

Figura 3.5: Estrutura da rede convolucional. Fonte: [3].

Além disso, foram utilizados vários cenários de treinamento da rede usando diferentes níveis de aumento de dados (do inglês: *data augmentation*) e pré-processamento de dados:

- Converter as imagens de entrada, de formato RGB de para a escala de cinza.
- Manter as imagens de entrada no formato RGB.
- Converter as imagens de entrada, de formato RGB para o formato HSV.
- Converter as imagens de entrada, de formato RGB para o formato HSV e para a escala de cinza, e além disso, combiná-los.

- Aplicar alterações aleatórias de tonalidade e saturação nas imagens de entrada no formato RGB, além disso, inverter aleatoriamente as imagens na horizontal e na vertical, e depois convertê-las para o formato HSV e para a escala de cinza, e combiná-los.

Na Figura 3.6 encontram-se os resultados utilizando os diferentes cenários. Pode-se perceber a alta acurácia nos resultados, porém o *dataset* em questão, possui imagens muito semelhantes entre o conjunto de teste e o conjunto de treinamento, como pode ser visto na Figura 3.7 podendo ter ocorrido um *overfitting*. Isso acarreta em um baixo poder de generalização do modelo proposto, isto é, o modelo pode não possuir a habilidade de realizar a classificação com tamanha acurácia em novos dados não apresentados durante o treinamento.

Scenario	Accuracy on training set	Accuracy on test set
Grayscale	99.96%	94.24%
RGB	99.23%	93.47%
HSV	99.98%	97.01%
HSV + Grayscale	99.78%	95.71%
HSV + Grayscale + hue/saturation change + flips	99.86%	97.04%

Figura 3.6: Resultados apresentados em [3], utilizando diferentes cenários de pré-processamento das imagens. Fonte: [3].

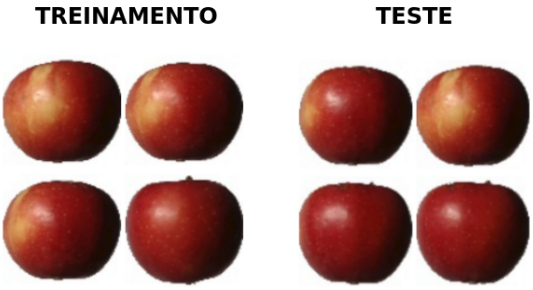

Figura 3.7: Semelhança entre o conjunto de treinamento e de teste no *dataset Fruits-360*. Fonte: Produzido pelo autor.

A partir desses estudos, é interessante notar como a rede neural convolucional consegue fazer o trabalho de extrair características das

imagens e produzir resultados esperados. Porém, para isso, é necessário escolher sabiamente o conjunto de dados que será utilizado e a estrutura da rede neural, para que haja boa convergência dos resultados e que a rede tenha um robusto poder de generalização.

CAPÍTULO 4

Método Proposto

Como o *dataset* utilizado possui imagens de frutas com diversos planos de fundo, é necessário uma metodologia adequada de análise e treinamento das redes neurais, e também dos testes de desempenho, para que os resultados encontrados sejam os melhores possíveis. Assim, o presente trabalho foi dividido em três etapas: análise e ampliação do *dataset*, treinamento e aplicação dos modelos de redes neurais e avaliação dos resultados encontrados.

A primeira etapa consiste na análise do *dataset* adquirido e também sua ampliação, para que as redes neurais tenham um número mínimo de imagens que garanta os resultados esperados. A etapa seguinte consiste na escolha, treinamento e aplicação dos modelos de redes neurais. Por fim, é realizada a análise dos resultados ao se utilizar os modelos treinados com o conjunto de teste. Nas próximas seções, serão detalhadas e justificadas todas essas etapas.

4.1 *Dataset*

Uma parte crucial presente nos trabalhos que utilizam aprendizado de máquina supervisionado, é a correta escolha ou construção do *dataset*.

Geralmente, o conjunto de dados deve ser dividido, no mínimo, em dois conjuntos, um para treinamento, e outro para avaliação de desempenho do método, o conjunto de teste. A escolha dos conjuntos deve ser feita cuidadosamente a fim de evitar problemas de *overfitting*, isto é, quando um modelo não consegue generalizar os detalhes e ruídos aprendidos durante o treinamento, afetando negativamente o desempenho do modelo em novos dados. Isso pode ocorrer quando o *dataset* de treinamento não possui amostras suficientes para representar a maioria dos casos presentes na vida real.

Nesse trabalho, serão usados dois *datasets*: o *dataset* presente em [1], cujo nome nesse trabalho será *dataset original*, e esse mesmo *dataset* ampliado, cujo nome será *dataset ampliado*. O objetivo na utilização desses dois conjuntos de dados, é realizar uma comparação dos resultados ao utilizar redes convolucionais com o resultado apresentado em [1]. O conjunto de teste permanecerá o mesmo, a fim de permitir comparações justas entre os métodos. Para um treinamento correto, uma parte do conjunto de treinamento será utilizada como conjunto de validação. O conjunto de validação é necessário para que haja uma avaliação imparcial durante o treinamento da rede, com o objetivo de melhorar a performance e ajustar os hiper parâmetros do modelo, sem que precise utilizar o conjunto de teste.

Geralmente um bom conjunto de validação possui uma quantidade semelhante de exemplares em relação ao conjunto de teste. Além disso, também é aconselhável que o conjunto de treinamento possua mais exemplares do que o conjunto de validação. Neste caso, em algumas classes de objetos, não há um numero suficiente de exemplares para que haja essa divisão e que reste uma quantidade suficiente no conjunto de treinamento. Assim, nesse trabalho, o tamanho o conjunto de validação será o valor mínimo entre 30% da quantidade do conjunto de treinamento e a quantidade do conjunto de teste, para cada classe. No processo de ampliação do *dataset* manteve-se esse mesmo método de divisão de conjuntos.

4.1.1 Obtenção das imagens

O *dataset* presente em [1] pode ser encontrado em [16], e ele foi explicado na Seção 3.1. Na Tabela 4.1 pode ser visualizado o *dataset original* dividido em conjuntos de treinamento, validação e teste.

Classe do objeto	Treinamento	Validação	Teste
Background	690	50	50
Morango	400	50	50
Pêssego	384	50	50
Laranja	382	50	50
Maça Vermelha	368	50	50
Tomate	361	50	50
Banana	344	50	50
Kiwi	261	50	50
Pera	218	50	50
Romã	216	50	50
Melancia	189	50	50
Abacaxi	175	50	50
Uva	115	49	50
Carambola	113	48	50
Limão	105	44	50
Manga	105	44	50
Jaca	84	36	50
Pomelo	83	35	50
Melão	72	30	50
Mamão	42	17	50
Total	4707	903	1000

Tabela 4.1: *Dataset original* separado em conjuntos de treinamento, validação e teste. Fonte: Produzido pelo autor.

Para o procedimento de aumento do *dataset original*, foi usada a biblioteca em *Python* chamada *Google Images Download*. Essa biblioteca funciona como um programa *Python* que pesquisa palavras-chave no *Google Imagens* e, opcionalmente, pode realizar o *download* das imagens resultantes para o seu computador.

A biblioteca utilizada pode realizar a pesquisa de imagens utilizando um arquivo de configuração que possui as palavras-chaves que serão pesquisadas. Neste trabalho, o formato do arquivo escolhido foi o *JSON* e foram escolhidas palavras-chave que representam o conjunto das 20 categorias de objetos escolhidos.

Após o armazenamento de todas as imagens pesquisadas, foi verificado que a maioria das imagens não tinham a ver com o assunto

pesquisado ou que possuía uma coleção de objetos, em vez de um único objeto específico. Assim, foi feito manualmente a seleção das imagens que poderiam enriquecer o *dataset*. Por fim, o *dataset ampliado* pode ser visto na Tabela 4.2.

Classe do objeto	Treinamento	Validação	Teste
Background	690	50	50
Morango	400	50	50
Pêssego	400	50	50
Laranja	400	50	50
Banana	400	50	50
Maça Vermelha	399	50	50
Tomate	397	50	50
Abacaxi	374	50	50
Pera	308	50	50
Kiwi	276	50	50
Romã	251	50	50
Melancia	211	50	50
Uva	209	50	50
Manga	173	50	50
Carambola	171	50	50
Mamão	164	50	50
Limão	142	50	50
Melão	138	50	50
Jaca	120	50	50
Pomelo	117	50	50
Total	5740	1000	1000

Tabela 4.2: *Dataset* utilizado em [1] ampliado. Fonte: Produzido pelo autor.

4.1.2 Balanceamento do *dataset*

Analisando as tabelas 4.1 e 4.2, pode-se notar que a distribuição de exemplares por classe nos conjuntos de treinamentos está desbalanceada. Dados desbalanceados são bem comuns, principalmente em conjuntos de dados reais, onde há sempre algum grau de desequilíbrio. Se o nível de desequilíbrio for relativamente baixo, é possível que não tenha nenhum grande impacto no desempenho do modelo que será treinado.

Porém, em conjuntos de dados muito desbalanceados, como o caso dos *datasets* utilizados nesse trabalho, os métodos podem apresentar uma maior probabilidade de classificar novas observações para a classe majoritária.

Para combater esse desbalanceamento há diversas técnicas, como por exemplo: conseguir mais dados de classes minoritárias, fazer uma re-amostragem do conjunto de dados, usar geradores de amostras sintéticas ou também penalizar o treinamento do modelo. Nesse trabalho foi escolhido o último caso, pois na biblioteca *Keras* já existe um atributo que realiza essa penalização. O atributo chamado *class_weight* relaciona o índice da classe com um peso específico, durante o treinamento do modelo. Isso serve para forçar o modelo a "prestar mais atenção" nas amostras das classe sub-representadas.

4.1.3 *Data Augmentation*

Além de balancear as classes durante o treinamento, uma outra técnica que ajuda na melhoria do modelo é o acréscimo de dados (do inglês: *data augmentation*). No mundo real, conjunto de imagens pode ter sido obtido a partir de um conjunto limitado de condições. Porém, é necessário que o modelo treinado possa realizar classificações em uma variedade de condições, tais como orientação, localização, escala, brilho, etc. Para isso, usa-se *data augmentation* para treinar a rede neural com dados adicionais modificados sinteticamente.

A biblioteca *Keras* fornece a capacidade de *data augmentation* através da classe *ImageDataGenerator*. Uma variedade de técnicas é suportada, assim como métodos de escalonamento de *pixels*. A seguir estão algumas técnicas que podem ser usadas:

- Deslocamento da imagem (usando os argumentos *width_shift_range* e *height_shift_range*).
- Inversão da imagem (usando os argumentos *horizontal_flip* e *vertical_flip*).
- Rotações da imagem (usando o argumento *rotation_range*).
- Brilho da imagem (usando o argumento *brightness_range*).
- Zoom da imagem (usando o argumento *zoom_range*).

Ao usar o *ImageDataGenerator*, as imagens do conjunto de dados não são usadas diretamente. Em vez disso, apenas imagens transformadas são fornecidas ao modelo, onde o tipo de transformação realizada é aleatória. As imagens transformadas não ficam salvas, isto é, a cada iteração é gerado um novo lote de imagens alteradas para o treinamento do modelo. Na Figura 4.1, podem ser vistas algumas transformações realizadas na imagem original.

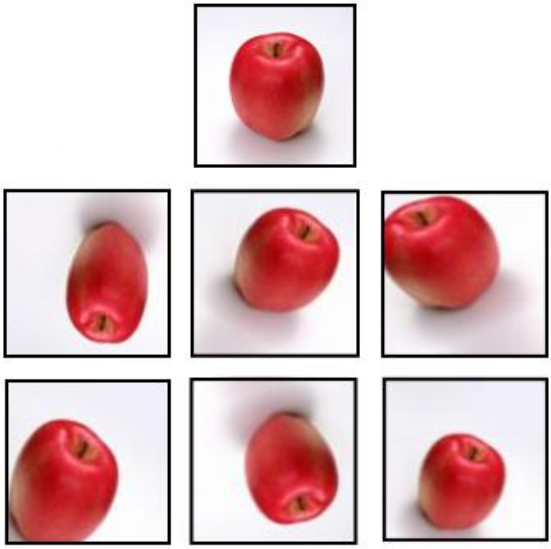

Figura 4.1: Exemplos de *data augmentation*, a imagem original se encontra no topo. Fonte: Produzido pelo autor.

4.2 Escolha das Redes Neurais

Para a realização do trabalho foram escolhidos alguns modelos pré-treinados disponíveis na biblioteca *Keras* [20], e também a rede convolucional presente em [3]. Os modelos pré-treinados já foram exaustivamente treinados em grandes conjuntos de dados, obtendo resultados surpreendentes, como mostra a Tabela 4.3. Nessa tabela constam as redes escolhidas para esse trabalho. Essas redes fazem parte das redes vencedoras de competições como o ILSVRC e a Tabela 4.3 apresenta

suas respectivas acurácias *top-1* e *top-5* no *dataset* de validação do *ImageNet*. Além disso, é possível notar a profundidade e a quantidade de parâmetros de cada rede.

Modelo	Tamanho (MB)	Acurácia Top-1 (%)	Acurácia Top-5 (%)	Parâmetros	Profundidade
Xception	88	79.0	94.5	22,910,480	126
InceptionV3	92	77.9	93.7	23,851,784	159
VGG16	528	71.3	90.1	138,357,544	23
VGG19	549	71.3	90.0	143,667,240	26
MobileNetV2	14	71.3	90.1	3,538,984	88

Tabela 4.3: Redes neurais e suas respectivas acurácias no *dataset ImageNet*. Fonte: Adaptado de [20].

As redes foram escolhidas de acordo com suas características para que fosse possível analisar os resultados em diferentes configurações de profundidade da rede, quantidade de parâmetros e tamanho do modelo. Nas seções a seguir, cada modelo é brevemente explicado.

4.2.1 VGG

O VGG16 e o VGG19 são modelos de redes neurais convolucionais propostos em [21]. Os modelos atingiram bons resultados em 2014 e com isso ficaram famosos em função do ILSVRC-2014. Os modelos realizaram uma melhoria nos modelos usados anteriormente, ao substituir o tamanho de filtros grandes por múltiplos filtros de tamanho 3×3, um após o outro. De acordo com [21], esse modelo se generaliza bem para uma ampla gama de tarefas e conjuntos de dados, e confirma a importância de utilizar redes profundas em classificações visuais. As arquiteturas dos dois modelos podem ser vistas na Figura 4.2.

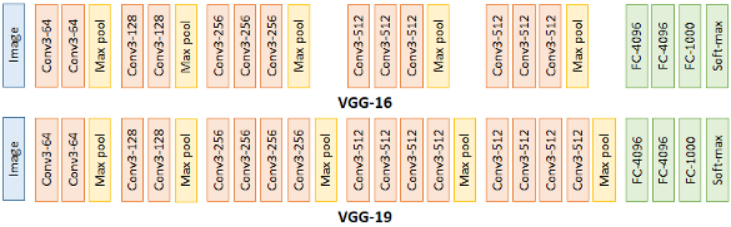

Figura 4.2: Estrutura dos modelos VGG16 e VGG19. Fonte: [22].

Infelizmente, existem duas grandes desvantagens com esses mode-

los. Devido às suas profundidades e ao número de nós totalmente conectados, o tamanho das redes é grande. Isso torna a implantação delas em sistema embarcados uma tarefa problemática. Além disso, o treinamento da rede é demorado, devido à quantidade de parâmetros presentes.

4.2.2 InceptionV3

O objetivo do módulo da rede Inception é atuar como um "extrator de características de vários níveis", realizando convoluções 1×1, 3×3 e 5×5 dentro do mesmo módulo da rede [23]. A saída desses filtros é então concatenada de acordo com a dimensão do canal, antes de entrar na próxima camada da rede, como mostra a Figura 4.3.

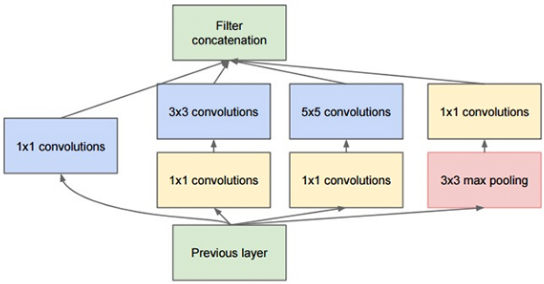

Figura 4.3: Módulo Inception. Fonte: [24].

Como a rede Inception é bem profunda, ela é computacionalmente complexa. Assim foi adicionada uma camada extra de convolução 1×1 antes das convoluções de 3×3 e 5×5 para limitar o número de canais de entrada. Além disso, a convolução 5×5 foi fatorada para duas operações de convolução 3×3 para melhorar a velocidade computacional. Esse empilhamento de duas convoluções 3×3 leva a um aumento no desempenho do modelo. A estrutura da rede Inception pode ser vista na Figura 4.4, onde também é possível notar os módulos Inception e a profundida grande da rede.

4.2.3 Xception

Xception [26] é uma extensão da arquitetura Inception que substitui os módulos Inception por convoluções pontuais seguidas de convoluções

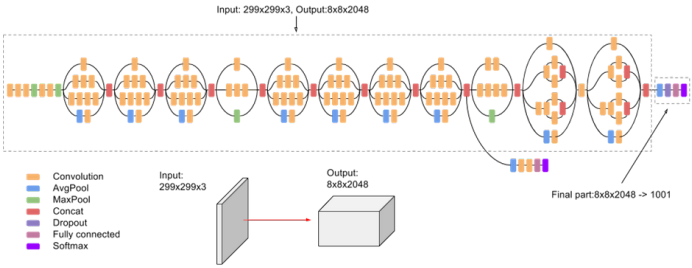

Figura 4.4: Estrutura da Rede Inception. Fonte: [25].

em profundidade, como mostra a Figura 4.5.

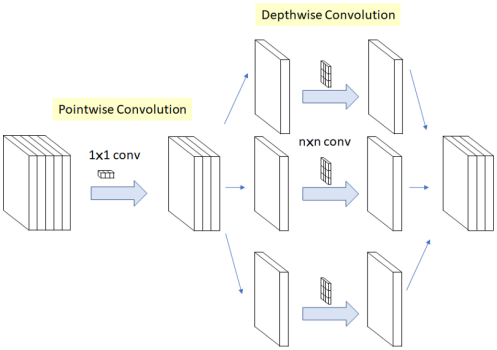

Figura 4.5: Modelo de convoluções pontuais seguidas de convoluções em profundidade. Fonte: [27].

A convolução em profundidade é a convolução espacial de tamanho $n \times n$ no sentido do canal do modelo. Em comparação com a convolução convencional, não é necessário executar a convolução em todos os canais, reduzindo o número de conexões e tornando o modelo mais leve. A rede Xception com essas configurações possui melhores resultados no *ImageNet* do que as redes VGG e a InceptionV3.

4.2.4 MobileNetV2

As redes MobileNet [28] [29], também possuem camadas de convolução em profundidade assim como a Xception. Como explicado anteriormente, essas camadas são usada para reduzir o tamanho e a complexi-

dade do modelo. Assim, faz com que essas redes sejam particularmente úteis para aplicativos e sistemas embarcados. Como pode ser visto na tabela 4.3, ela é a rede que possui o menor tamanho e a menor quantidade de parâmetros, mas suas acurácias não estão distantes dos melhores modelos.

Para obter essas características menores, dois parâmetros são introduzidos para que a rede MobileNet possa ser facilmente utilizada: o multiplicador de largura (α) e o multiplicador de resolução (ρ). O multiplicador de largura é usado para controlar a largura de entrada da camada. Seu valor é entre 0 e 1, e o custo computacional e o número de parâmetros podem ser reduzidos quadraticamente por aproximadamente α^2. Já o multiplicador de resolução é introduzido para controlar a resolução da imagem de entrada da rede, diminuindo também o número de parâmetros.

As redes MobileNet provam que é possível obter bons modelos de redes neurais para que sejam aplicados em sistemas embarcados, que é a tendência tecnológica presente hoje.

4.3 Configuração das Redes Neurais

Poucos trabalhos treinam uma rede convolucional inteira para um único projeto, pois é relativamente raro ter um conjunto de dados de tamanho suficiente e também exige um grande custo computacional. Em vez disso, é comum usar modelos pré-treinados em um conjunto de dados muito grande (como explicado na Seção 2.2.5) e usar esses modelos como inicializações ou como extratores de características para a tarefa de interesse. Esse conceito se chama transferência de aprendizado (do inglês: *Transfer Learning*), em vez de iniciar o processo de aprendizado do início, pode-se começar a partir de padrões que foram aprendidos ao resolver um problema diferente. Transferência de aprendizagem é um método popular em visão computacional e aprendizado de máquina porque permite construir modelos precisos de uma maneira que economiza tempo e *hardware*[8].

4.3.1 Transferência de aprendizagem

De acordo com o explicado na Seção 2.2.2, uma rede convolucional pode ser descrita em duas partes: uma base convolucional e um classificador,

como mostrado na Figura 4.6. A base convolucional tem como principal objetivo extrair características das imagens, e o classificador utiliza essas características para corretamente classificar a classe pertencente à imagem.

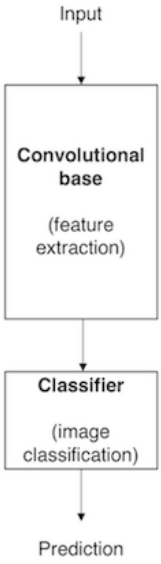

Figura 4.6: Arquitetura básica de uma rede convolucional. Fonte: [30].

Utilizando o conceito de CNN descrito acima e também de transferência de aprendizagem, é comum seguir a seguinte orientação para utilizar um modelo pré-treinado: primeiro é removido o classificador do modelo, depois adiciona-se o classificador que se encaixa no projeto e por fim fazem-se ajustes no modelo em busca de uma melhor performance. Esses ajustes podem seguir três estratégias:

- Treinar o modelo inteiro, isto é, utilizar a arquitetura do modelo pré-treinado e treiná-lo de acordo com o conjunto de dados do projeto. Para isso é necessário um grande conjunto de dados e um grande poder computacional.
- Treinar algumas camadas da rede e deixar as outras "congeladas". As camadas iniciais aprendem características muito gerais e à medida que se avança na rede, as camadas tendem a aprender

padrões mais específicos para a tarefa em que está sendo treinada. Assim, para este caso, pode-se deixar as camadas iniciais "congeladas" e treinar novamente as camadas finais de acordo com o projeto. Esse método é eficiente quando o conjunto de dados é pequeno.

- "Congelar" toda a base convolucional e treinar somente o classificador. Neste caso, o modelo pré-treinado servirá como um extrator de características fixo para o conjunto de dados do projeto. Pode ser útil se houver pouco poder computacional, se o conjunto de dados for pequeno ou se o modelo pré-treinado resolver um problema semelhante ao do projeto.

Para este trabalho, serão utilizadas essas três estratégias. A primeira será utilizada na rede neural convolucional presente em [3]. As duas últimas estratégias serão utilizadas principalmente nos modelos pré-treinados, já que o conjunto de dados não possui muitos exemplares e o poder computacional é limitado.

4.4 Método de avaliação de desempenho

Após o treinamento dos modelos, os pesos e a estrutura de cada modelo são salvos em um arquivo que será utilizado para avaliar a eficiência do modelo no conjunto de testes.

Como forma de avaliar o desempenho de todos os modelos usados nesse trabalho, é necessária uma medida que permita comparações com outras metodologias. A partir dos trabalhos descritos na Seção 3, é possível notar que a medida de acurácia no conjunto de teste é a mais utilizada para problemas de classificação. Nas redes utilizadas, os modelos fazem uma previsão utilizando uma imagem em questão e obtém como resultado uma saída probabilística, isto é, a probabilidade daquela imagem pertencer a cada classe. Assim, no caso da acurácia *top-1*, é verificado se a classe superior (a que tem a maior probabilidade) é a mesma que a classe correta da imagem. No caso da acurácia *top-5*, é verificado se classe correta está entre as cinco principais previsões (as cinco com as maiores probabilidades).

Na Figura 4.7, mostra-se um exemplo de predições para a classe "maçã". Como pode ser visto, a classe "maçã" possui a maior probabi-

lidade, contabilizando a acurácia *top-1*, e também a classe está presente entre as cinco maiores predições, contabilizando a acurácia *top-5*.

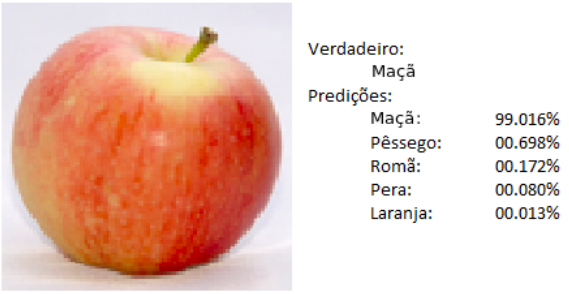

Figura 4.7: Exemplo de predições. Fonte: Produzido pelo autor.

4.5 *Software* de Processamento

O treinamento de redes neurais profundas necessita de um potente poder computacional devido à grande complexidade dos modelos utilizados, principalmente ao lidar com imagens. Atualmente, os modelos de estado da arte utilizam processadores e placas de vídeo, que possuem centenas de unidades de processamento e aceleram muito o processo de aprendizado das redes. Um maquinário desse tipo é escasso e necessita de um alto custo para sua obtenção. Assim, foi optado por realizar o treinamento e avaliação das redes neurais na ferramenta chamada *Google Colab* [31].

O *Google Colab* é um serviço na nuvem gratuito que oferece suporte de processamento, nesse caso, uma GPU *Tesla K80*. Por causa disso, ele é uma ótima ferramenta para o treinamento de algoritmos de aprendizado de máquina. Como comparação, o treinamento do modelo *VGG16* usando transferência de aprendizagem, em um computador pessoal sem placa de vídeo dedicada, demorou cerca de 680 segundos por época. Ao utilizar o *Google Colab* com essa mesma rede, o tempo de treinamento foi de aproximadamente 16 segundos por época (42,5 vezes mais rápido). Nota-se com esse resultado, a importância de utilizar um *hardware* dedicado em tarefas que exigem muito poder computacional, pois reduzindo o tempo de processamento, pode-se treinar redes ainda mais profundas e com mais parâmetros que influenciam no resultado.

CAPÍTULO 5

Resultados

Neste capítulo, os resultados das diferentes soluções propostas no trabalho são apresentados. Primeiramente é realizada uma comparação dos resultados obtidos utilizando os métodos escolhidos, a fim de escolher o método que obteve os melhores resultados. Em seguida são realizados simulações com o melhor método encontrado, para uma busca de melhorias no modelo com o objetivo de conseguir alavancar o resultado.

5.1 Resultados dos modelos escolhidos

Nesta seção serão apresentados os resultados dos modelos escolhidos na Seção 4.2 utilizando o *dataset original*. A configuração de *data augmentation*, taxa de aprendizado e o otimizador se mantiveram os mesmos em todos os treinamentos. O otimizador escolhido foi o *RMSprop* com uma taxa de aprendizado fixa de 0.0001. Além disso, o treinamento funcionou do seguinte modo: o modelo treina por até 100 épocas, utilizando o método *EarlyStopping*. Isto é, o método *EarlyStopping* verifica se o valor da função de perda no conjunto de validação não melhorou após seis épocas consecutivas e interrompe o treinamento. Para o modelo ser reutilizado sem precisar realizar um novo treinamento, o método

ModelCheckpoint salva os pesos do modelo da época que conseguiu o menor valor da função de perda no conjunto de validação.

5.1.1 Rede Neural Convolucional

Antes de utilizar os modelos pré-treinados, foram feitas simulações utilizando o treinamento de uma rede convolucional inteira. A rede em questão foi a mesma utilizada em [3]. Ela é composta por cinco camadas convolucionais, com filtros de tamanho 5×5, seguidas de camadas *max pooling*. Por fim, existem duas camadas totalmente conectadas e a camada de classificação.

Primeiramente foi realizado o treinamento e avaliação dessa rede com o seu próprio conjunto de dados (*Fruits-360*) e com sua configuração, explicados na Seção 3.2, para verificar se a rede funciona adequadamente. Na tabela 5.1 estão os resultados obtidos, que se aproximam dos resultados encontrados em [3].

Rede	Épocas	Duração do Treinamento	Acurácia Top-1 (%)	Acurácia Top-5 (%)	Total de Parâmetros	Parâmetros Treináveis
CNN *Fruits-360*	75	2.19 horas	98.93	99.98	5,273,073	5,273,073

Tabela 5.1: Resultado da rede convolucional com o *dataset Fruits-360*. Fonte: Produzido pelo autor.

Após isso, foi realizado o treinamento e avaliação da mesma rede convolucional utilizando o *dataset original*, presente em [1], e o *dataset ampliado*. Pelos resultados presentes na tabela 5.2, é possível notar a diferença na duração do treinamento em relação à tabela 5.1, devido à diferença de tamanhos dos conjuntos de dados. Além disso, nota-se uma acurácia top-1 menor comparada ao resultado anterior.

Rede	Épocas	Duração do Treinamento	Acurácia Top-1 (%)	Acurácia Top-5 (%)	Total de Parâmetros	Parâmetros Treináveis
CNN *dataset original*	63	0.28 horas	74.40	95.70	5,273,073	5,273,073
CNN *dataset ampliado*	72	0.36 horas	68.10	96.20	5,257,396	5,257,396

Tabela 5.2: Resultados da rede convolucional com o *dataset original* e com o *dataset ampliado*. Fonte: Produzido pelo autor.

Por fim, utilizando o conceito de *transfer learning*, foi utilizado modelo treinado no *dataset Fruits-360* para realizar a classificação do *da-*

taset original e do *dataset ampliado*. Isto é, a rede e seus pesos foram mantidos, e o classificador foi alterado para se encaixar nos *datasets*. O classificador manteve a mesma configuração e somente sua saída foi alterada, para condizer com a classificação de 20 classes dos *datasets*. Ao analisar a tabela 5.3, nota-se uma queda nas acurácias ao se utilizar *transfer learning*. Isso demonstra que as características aprendidas com o *dataset Fruits-360*, não se encaixaram corretamente com os *datasets*, isto é, o modelo treinado não possuía a capacidade de generalizar. Assim, é possível perceber que o *dataset Fruits-360* não representa satisfatoriamente a diversidade esperada para que o modelo tivesse um alto poder de generalização.

Rede	Épocas	Duração do Treinamento	Acurácia Top-1 (%)	Acurácia Top-5 (%)	Total de Parâmetros	Parâmetros Treináveis
CNN *transfer learning dataset original*	31	0.14 horas	70.2	95.2	5,010,356	4,740,116
CNN *transfer learning dataset ampliado*	45	0.29 horas	72.2	96.0	5,010,356	4,740,116

Tabela 5.3: Resultados da rede convolucional com o *dataset original* e com o *dataset ampliado* utilizando *transfer learning*. Fonte: Produzido pelo autor.

5.1.2 Modelos pré-treinados

Após as simulações com a rede convolucional, foi realizado o treinamento e testes dos modelos pré-treinados. Do mesmo modo, somente as últimas camadas foram trocadas, isto é, o classificador, que foi o mesmo utilizado na simulação com *transfer learning* da rede convolucional anterior. Nas tabelas 5.4 e 5.5, pode-se visualizar os resultados obtidos com o *dataset original* e com o *dataset ampliado*, respectivamente. Pelos resultados é possível perceber que a rede que obteve a maior acurácia foi a rede VGG16, e que as outras redes mais profundas ou mais leves não produziram um resultado satisfatório. Na Figura 5.1 encontram-se as curvas da acurácia e da perda construídas durante o treinamento da rede. Além disso, é possível notar que nessas simulações e nas passadas, ao se utilizar o *dataset ampliado*, não houve uma melhora esperada nos resultados.

Rede	Épocas	Duração do Treinamento	Acurácia Top-1 (%)	Acurácia Top-5 (%)	Total de Parâmetros	Parâmetros Treináveis
VGG16	37	0.16 horas	89.3	99.0	19,701,844	4,987,156
VGG19	38	0.17 horas	87.5	98.9	25,011,540	4,987,156
MobileNetV2	17	0.08 horas	68.3	90.9	23,498,068	21,240,084
Xception	16	0.07 horas	58.4	86.2	40,004,412	19,142,932
InceptionV3	24	0.11 horas	57.9	89.1	24,168,500	2,365,716

Tabela 5.4: Resultados dos modelos pré-treinados com o *dataset original* utilizando *transfer learning*. Fonte: Produzido pelo autor.

Rede	Épocas	Duração do Treinamento	Acurácia Top-1 (%)	Acurácia Top-5 (%)	Total de Parâmetros	Parâmetros Treináveis
VGG16	37	0.31 horas	89.4	98.6	19,701,844	4,987,156
VGG19	31	0.28 horas	86.1	98.7	25,011,540	4,987,156
MobileNetV2	15	0.11 horas	67.7	95.3	23,498,068	21,240,084
Xception	15	0.13 horas	57.3	88.9	40,004,412	19,142,932
InceptionV3	21	0.17 horas	55.7	85.0	24,168,500	2,365,716

Tabela 5.5: Resultados dos modelos pré-treinados com o *dataset ampliado* utilizando *transfer learning*. Fonte: Produzido pelo autor.

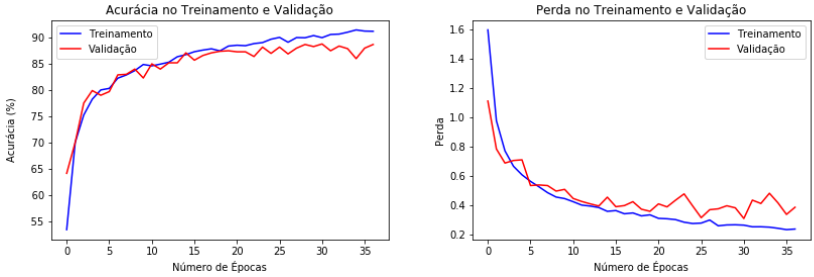

Figura 5.1: Curvas da acurácia e da função perda durante o treinamento do modelo VGG16 com o *dataset original*. Fonte: Produzido pelo autor.

5.2 Simulações com o melhor método

Utilizando o modelo VGG16, que obteve o melhor resultado nas simulações anteriores, foram realizados testes com diferentes configurações dessa mesma rede com o objetivo de melhorar a acurácia do modelo. Todas as simulações a seguir foram feitas utilizando o *dataset original*,

já que o *dataset ampliado* não provocou uma melhora nos resultados. Para realizar essa busca de melhores configurações, a acurácia das simulações foi obtida no conjunto de validação, para reduzir a chance de causar *overfitting* do modelo. Primeiramente, foram feitas simulações com outros dois otimizadores, mantendo a taxa de aprendizado ainda fixa, no valor de 0.0001. Pelo resultado da tabela 5.6, o otimizador *RMSprop*, foi o que obteve uma melhor acurácia.

Otimizador	Épocas	Duração do Treinamento	Acurácia Top-1 (%)	Acurácia Top-5 (%)
RMSprop	27	0.13 horas	89.7	98.7
Adam	26	0.13 horas	87.6	98.5
SGD	100	0.55 horas	63.0	86.1

Tabela 5.6: Resultados do modelo VGG16 utilizando diferentes otimizadores, com o conjunto de validação do *dataset original*. Fonte: Produzido pelo autor.

Após a descoberta do melhor otimizador, foi feito o estudo da melhor configuração da taxa de aprendizado. Dois métodos de redução da taxa de aprendizado foram utilizados: *LearningRateScheduler* e *ReduceLROnPlateau*. O método *LearningRateScheduler* utiliza uma função que reduz a taxa de aprendizado por um fator específico a cada época do treinamento. Já o método *ReduceLROnPlateau* reduz a taxa de aprendizado por um fator específico, uma vez que o aprendizado se estagna, isto é, se a cada duas épocas, o valor da função de perda no conjunto de validação não melhorar, a taxa de aprendizado é reduzida. Os resultados são mostrados na tabela 5.7, onde é possível notar que o único método que obteve uma acurácia melhor do que utilizando uma taxa de aprendizado fixa, foi o *ReduceLROnPlateau* com fator de redução de 0.5.

Método	Épocas	Duração do Treinamento	Acurácia Top-1 (%)	Acurácia Top-5 (%)
ReduceLROnPlateau fator 0.5	26	0.12 horas	91.4	99.4
ReduceLROnPlateau fator 0.9	24	0.11 horas	89.6	98.9
LearningRateScheduler fator 0.5	33	0.15 horas	84.3	96.2
LearningRateScheduler fator 0.9	27	0.20 horas	89.6	98.5

Tabela 5.7: Resultados com o otimizador *RMSprop* no conjunto de validação utilizando diferentes métodos de redução da taxa de aprendizado. Fonte: Produzido pelo autor.

Com os resultados de melhor otimizador e da melhor configuração da taxa de aprendizado, foram realizadas simulações com mais um método de *transfer learning* e mais um tipo de classificador, além dos que já foram utilizados. Todos eles são explicados abaixo:

- Classificador 1: O classificador que já estava sendo usado nas simulações anteriores. Ele é o mesmo utilizado em [3], e possui duas camadas totalmente conectadas com 1024 e 256 unidades, com função de ativação ReLU. Por fim, ele possui uma ultima camada com função de ativação *softmax* que realiza a classificação das 20 classes.

- Classificador 2: Esse classificador possui a mesma configuração das últimas camadas da rede VGG16, isto é, ele possui três camadas totalmente conectadas com 4096, 4096 e 1000 unidades, com função de ativação ReLU. Por fim, ele possui uma ultima camada com função de ativação *softmax* que realiza a classificação das 20 classes.

- *Transfer learning* 1: "Congela" a base convolucional da rede VGG16 e deixa somente o classificador para ser treinado com o conjunto de dados.

- *Transfer learning* 2: Treina as quatro últimas camadas da base convolucional da rede VGG16 e deixa o restante "congelado". As

quatro últimas camadas se referem a três camadas convolucionais e uma camada de *max pooling*. Nesse método o classificador também continua sendo treinado com o conjunto de dados.

Com os resultados da tabela 5.8, verifica-se que o segundo método de *transfer learning* e o segundo classificador provocaram o melhor resultado no conjunto de validação do *dataset original*. Essas configurações representam o treinamento das últimas 8 camadas da rede VGG16, deixando as 15 camadas anteriores congeladas.

Configuração	Épocas	Duração do Treinamento	Acurácia Top-1 (%)	Acurácia Top-5 (%)	Total de Parâmetros	Parâmetros Treináveis
Transfer learning 1 Classificador 1	26	0.12 horas	91.4	99.4	19,701,844	4,987,156
Transfer learning 1 Classificador 2	26	0.15 horas	92.1	99.3	50,456,404	35,741,716
Transfer learning 2 Classificador 1	31	0.20 horas	94.8	99.8	19,701,844	12,066,580
Transfer learning 2 Classificador 2	25	0.18 horas	95.2	99.9	50,456,404	42,821,140

Tabela 5.8: Resultados do modelo VGG16 no conjunto de validação utilizando dois classificadores e dois métodos de *transfer learning*. Fonte: Produzido pelo autor.

Assim foi encontrada a melhor configuração da rede VGG16 para o *dataset original*: método de *transfer learning* 2, Classificador 2, método *ReduceLROnPlateau* com fator de redução de 0.5 e utilizando o otimizador *RMSprop*. Após isso, foram realizadas simulações com essa melhor configuração nos conjuntos de teste do *dataset original* e do *dataset ampliado*, com os resultados mostrados na tabela 5.9. Percebe-se que utilizando a melhor configuração foi possível aumentar em aproximadamente 4% a acurácia top-1 do modelo VGG16 com o *dataset original*. Na Figura 5.2 encontram-se as curvas da acurácia e da perda construídas durante o treinamento da rede.

Conjunto de Dados	Épocas	Duração do Treinamento	Acurácia Top-1 (%)	Acurácia Top-5 (%)	Total de Parâmetros	Parâmetros Treináveis
VGG16 *dataset original*	31	0.17 horas	93.4	99.5	50,456,404	42,821,140
VGG16 *dataset ampliado*	15	0.16 horas	92.8	99.5	50,456,404	42,821,140

Tabela 5.9: Resultados utilizando o classificador 2 e o método *transfer learning* 2 nos conjuntos de teste do *dataset original* e do *dataset ampliado*. Fonte: Produzido pelo autor.

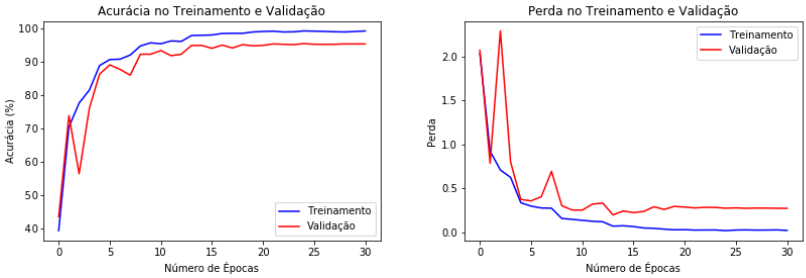

Figura 5.2: Curvas da acurácia e da função perda durante o treinamento do modelo VGG16 com o *dataset original* e com a melhor configuração. Fonte: Produzido pelo autor.

Por fim, foram realizadas duas últimas simulações utilizando o conceito de *transfer learning* de permitir o treinamento da rede inteira com o conjunto de dados, isto é, nenhuma camada foi congelada no treinamento. Além disso, foi utilizado essa configuração nos dois *datasets*, e os resultados nos conjuntos de testes são mostrados na tabela 5.10.

Configuração	Épocas	Duração do Treinamento	Acurácia Top-1 (%)	Acurácia Top-5 (%)	Total de Parâmetros	Parâmetros Treináveis
VGG16 *dataset original*	26	0.41 horas	91.9	99.2	50,456,404	50,456,404
VGG16 *dataset ampliado*	20	0.36 horas	93.2	99.6	50,456,404	50,456,404

Tabela 5.10: Resultados ao treinar toda a rede VGG16 com os *datasets original e ampliado*. Fonte: Produzido pelo autor.

5.3 Discussão dos resultados

A partir dos resultados encontrados, pode-se perceber a importância das redes pré-treinadas para o desenvolvimento de novos métodos para a classificação de imagens. Como essas redes foram treinadas em um grande conjunto de dados, possuem maiores profundidades e uma maior quantidade de parâmetros, elas conseguem aprender as características gerais das imagens com maior desempenho do que se a rede for construída com um conjunto de dados específico de tamanho reduzido.

Ao se utilizar a rede convolucional presente em [3] em seu próprio conjunto de dados (tabela 5.1), verificou-se uma grande acurácia. Porém ao utilizar a mesma rede em um novo conjunto de dados, ou utilizando *transfer learning* com os pesos originais, não foi possível obter uma acurácia próxima desse valor. Esse fato pode ter acontecido devido ao fato de que o conjunto de dados presente em [3] possui um conjunto de teste muito parecido com o conjunto de treinamento, favorecendo o aumento da acurácia e diminuindo o poder de generalização do modelo.

Utilizando as redes pré-treinadas foi possível observar a eficiência dos métodos de *transfer learning* (tabelas 5.4, 5.5 e 5.9). Com essas redes é possível utilizar um conjunto de dados de tamanho moderado para treinar algumas de suas camadas e com isso obter um modelo que possui uma acurácia relevante.

Também foi possível notar que ao treinar a rede toda com o conjunto de dados desse trabalho (tabela 5.10), a acurácia no *dataset ampliado* foi maior do que no *dataset original*, sugerindo que um aumento do conjunto de dados pode realmente melhorar a acurácia do modelo. Porém, alguns resultados mostraram uma redução na acurácia ao utilizar o *dataset ampliado*. Pode-se explicar esse resultado, levando em conta que foram adicionados mais exemplares com planos de fundo diferentes e com diferentes posições dos objetos na imagem. Isso levou o modelo a cometer alguns erros a mais em classes que já estavam com bons desempenhos.

Por fim, em comparação com o resultado apresentado em [1], o método utilizando a rede VGG16 com a melhor configuração (tabela 5.9), obteve uma acurácia superior ao se utilizar o mesmo conjunto de dados. Esse resultado indica que as redes convolucionais podem extrair características das imagens sem precisar de um pré-processamento da imagem antes da entrada da rede neural.

CAPÍTULO 6

Conclusão

Neste trabalho foi possível validar a possibilidade de utilização de aprendizado de máquina para a classificação de imagens de frutas, obtendo uma acurácia desejável. As propostas de implementação foram baseadas em redes neurais convolucionais, utilizando modelos pré-treinados e também a rede neural presente em [3].

Cada proposta apresenta suas especificidades e parâmetros, e com isso, mostra resultados diferentes. Os resultados apresentados no Capítulo 5 indicou que a proposta com melhor desempenho foi a que utilizou a rede VGG16, pré-treinada no conjunto de dados *ImageNet*. Esse fato corrobora a tendência de avanço do uso das técnicas de aprendizagem profundas, especificamente de redes convolucionais, que tem sido observada na comunidade de processamento de imagens nos últimos anos.

Esse trabalho também demonstra que as técnicas de aprendizado de máquina não se restringem a grandes conjuntos de dados, mas que podem ser empregadas em situações com volume reduzido de amostras, inclusive desbalanceadas, ao se encontrar um modelo que se encaixa àquela tarefa específica.

Além disso, a utilização de *hardwares* potentes para o treinamento dessas redes impulsiona o desenvolvimento de algoritmos complexos,

proporcionando soluções que antigamente não eram possíveis. Por fim, este trabalho demonstrou a viabilidade de aplicação de técnicas de aprendizado de máquina para fornecer uma melhoria no sistema de classificação de frutas. Esta solução é importante, pois as tecnologias utilizadas podem auxiliar no desenvolvimento da indústria e contribuir para a inovação de tradicionais setores da economia.

6.1 Trabalhos futuros

A partir desse trabalho são sugeridas duas propostas de trabalhos futuros, que são interligadas. Primeiramente, a construção de um conjunto de dados que melhor represente o mundo real, com frutas de várias classes, diferentes planos de fundo, e com imagens com múltiplas frutas de uma mesma classe. Assim, poderia se utilizar métodos de segmentação da visão computacional juntamente com redes neurais convolucionais para que ocorra a correta classificação das imagens.

Além disso, a partir desse novo conjunto de dados, a segunda proposta é realizar a classificação de frutas em tempo real, com o objetivo de implementar esse sistema em caixas de atendimento das redes de supermercados. Esse processo iria possivelmente acelerar o processo de compras dos usuários, além de diminuir o tamanho das filas. Por limitação de tempo essa solução não pôde ser implementada até a apresentação desse trabalho.

Referências bibliográficas

[1] Hulin Kuang, Leanne Lai Hang Chan, Cairong Liu, and Hong Yan. Fruit classification based on weighted score-level feature fusion. *Journal of Electronic Imaging*, 01 2016.

[2] S. Jana, S. Basak, and R. Parekh. Automatic fruit recognition from natural images using color and texture features. In *2017 Devices for Integrated Circuit (DevIC)*, pages 620–624, 03 2017.

[3] Horea Mureșan and Mihai Oltean. Fruit recognition from images using deep learning. *Acta Universitatis Sapientiae, Informatica*, 10:26–42, 06 2018.

[4] José Gilberto Spasiani Rinaldi, Reinaldo Morabito, and Vilma Mayumi Tachibana. A importância da rapidez de atendimento em supermercados: um estudo de caso. *Gestão & Produção São Carlos*, 16(1):1–14, 03 2009.

[5] Wikipedia, the free encyclopedia. Amazon go. `https://en.wikipedia.org/wiki/Amazon_Go`. [Online; accessado em: 05 de Março de 2019].

[6] The Learning Machine. Convolutional neural network (cnn). `https://www.thelearningmachine.ai/cnn`. [Online; accessado em: 21 de Março de 2019].

[7] Wikimedia Commons. Hsv color solid cylinder. https://commons.wikimedia.org/wiki/File:HSV_color_solid_cylinder.png. [Online; accessado em: 22 de Março de 2019].

[8] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29:1–98, 06 2017.

[9] Maj Michal. Object detection and image classification with yolo. https://www.kdnuggets.com/2018/09/object-detection-image-classification-yolo.html. [Online; accessado em: 22 de Março de 2019].

[10] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 07 1959.

[11] Stylianos Kampakis. What deep learning is and isn't. https://thedatascientist.com/what-deep-learning-is-and-isnt/. [Online; accessado em: 23 de Março de 2019].

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[13] Dan Claudiu Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. pages 1237–1242, 07 2011.

[14] Wikipedia, the free encyclopedia. Overfitting. https://en.wikipedia.org/wiki/Overfitting. [Online; accessado em: 27 de Março de 2019].

[15] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. 05 2016.

[16] Hulin Kuang. Fruit dataset. https://www.researchgate.net/publication/283087342_Fruitdataset, 2015. [Online; accessado em: 27 de Março de 2019].

[17] Arivazhagan Selvaraj, Newlin Shebiah, Selva Nidhyananthan, and Lakshmanan Ganesan. Fruit recognition using color and texture features. *Journal of Emerging Trends in Computing and Information Sciences*, 1:90–94, 10 2010.

[18] National Space Development Agency of Japan (NASDA). Minimum distance classifier. http://sar.kangwon.ac.kr/etc/rs_note/rsnote/cp11/cp11-6.htm. [Online; accessado em: 21 de Maio de 2019].

[19] Israr Hussain, Qianhua He, and Zhuliang Chen. Automatic fruit recognition based on dcnn for commercial source trace system. *International Journal on Computational Science Applications*, 8:01–14, 06 2018.

[20] Keras. Keras applications. https://keras.io/applications/. [Online; accessado em: 10 de Março de 2019].

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 09 2014.

[22] McAI. Review of vggnet 1st runner up of ilsvlc 2014. https://mc.ai/paper-review-of-vggnet-1st-runner-up-of-ilsvlc/2014-image-classification/. [Online; accessado em: 21 de Abril de 2019].

[23] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[24] Adrian Rosebrock. Imagenet: Vggnet, resnet, inception, and xception with keras. https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/. [Online; accessado em: 21 de Abril de 2019].

[25] Google Cloud. Guia avançado do inception v3 no cloud tpus. https://cloud.google.com/tpu/docs/

inception-v3-advanced. [Online; accessado em: 22 de Abril de 2019].

[26] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[27] Tsang Sik-Ho. Review: Xception with depthwise separable convolution. https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution/-better-than-inception-v3-image-dc967dd42568. [Online; accessado em: 22 de Abril de 2019].

[28] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[29] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[30] Pedro Marcelino. Transfer learning from pre-trained models. https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751. [Online; accessado em: 01 de Maio de 2019].

[31] Google Colab. Introducing colaboratory. https://colab.research.google.com/notebooks/welcome.ipynb. [Online; accessado em: 10 de Maio de 2019].