UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Rio Claro - SP

ÉRIKA KAYOKO HAMAGUTI

CLASSIFICAÇÃO DO GRAU DE MATURIDADE DE MORANGOS A PARTIR DE IMAGENS USANDO CLASSIFICADORES SUPERVISIONADOS E REDES CONVOLUCIONAIS COMO EXTRATORAS DE CARACTERÍSTICAS

Rio Claro - SP

2024

UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"
Instituto de Geociências e Ciências Exatas
Campus de Rio Claro

ÉRIKA KAYOKO HAMAGUTI

CLASSIFICAÇÃO DO GRAU DE MATURIDADE DE MORANGOS A PARTIR DE IMAGENS USANDO CLASSIFICADORES SUPERVISIONADOS E REDES CONVOLUCIONAIS COMO EXTRATORAS DE CARACTERÍSTICAS

Dissertação de Mestrado apresentada ao Instituto de Geociências e Ciências Exatas do Campus de Rio Claro, da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos para obtenção do título de Mestre em Ciências da Computação.

Orientador: Prof. Dr. Fabricio Aparecido Breve

Rio Claro - SP

2024

H198c

Hamaguti, Érika Kayoko

Classificação do grau de maturidade de morangos a partir de imagens usando classificadores supervisionados e redes convolucionais como extratoras de características / Érika Kayoko Hamaguti. -- Rio Claro, 2024

81 p. : il., tabs., fotos

Dissertação (mestrado) - Universidade Estadual Paulista (UNESP), Instituto de Geociências e Ciências Exatas, Rio Claro

Orientador: Fabricio Aparecido Breve

1. Aprendizado do computador. 2. Inteligência artificial. 3. Morango. 4. Redes neurais (Computação). I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"
Instituto de Geociências e Ciências Exatas
Campus de Rio Claro

ÉRIKA KAYOKO HAMAGUTI

CLASSIFICAÇÃO DO GRAU DE MATURIDADE DE MORANGOS A PARTIR DE IMAGENS USANDO CLASSIFICADORES SUPERVISIONADOS E REDES CONVOLUCIONAIS COMO EXTRATORAS DE CARACTERÍSTICAS

Dissertação de Mestrado apresentada ao Instituto de Geociências e Ciências Exatas do Campus de Rio Claro, da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos para obtenção do título de Mestre em Ciências da Computação.

Comissão Examinadora

Prof. Dr. FABRICIO APARECIDO BREVE
IGCE / UNESP/Rio Claro (SP)

Prof. Dr. ZHAO LIANG
USP/São Paulo (SP)

Prof. Dr. DENIS HENRIQUE PINHEIRO SALVADEO
IGCE / UNESP/Rio Claro (SP)

Conceito: Aprovado.

Rio Claro (SP), 29 de julho de 2024.

AGRADECIMENTOS

Agradeço a Deus pela paciência e persistência concedidas para realizar esta jornada.

Agradeço aos meus pais, minha irmã, meus padrinhos de batizado e meus amigos por terem me apoiado e tornado possível obter esta conquista.

Agradeço ao meu orientador, Prof. Dr. Fabricio Aparecido Breve, por todos os conselhos e ensinamentos, pela ajuda e pela paciência nesse período.

RESUMO

Nos últimos anos, técnicas de Aprendizado de Máquina e Aprendizado Profundo têm sido aplicadas na pesquisa de morangos. Atualmente, identificar a maturidade dos morangos de forma eficiente e precisa é um desafio devido aos métodos tradicionais, que são baseadas na aparência ou composição química da fruta, serem caros e demorados. A classificação automática de morangos pode oferecer aos agricultores uma maneira mais precisa de avaliar a qualidade dos frutos. Este trabalho propõe a análise da eficiência de diferentes combinações de modelos supervisionados e Redes Neurais Convolucionais (CNNs) para classificar a maturidade dos morangos. Foram testadas 71 CNNs para extração de características das imagens, seguidas por uma redução de dimensionalidade com PCA e a aplicação de dez classificadores. Os melhores resultados foram obtidos com as CNNs da família ConvNeXt (ConvNeXtBase, ConvNeXtSmall e ConvNeXtTiny) e VGG (VGG16 e VGG19) em combinação com os classificadores *Gradient Boosting*, *Histogram Based Gradient Boosting* e SVM, alcançando acurácia acima de 72% e F1-Score acima de 78%. As combinações testadas demonstraram ser eficientes, proporcionando uma solução viável para a classificação precisa da maturidade dos morangos, potencialmente beneficiando os agricultores com uma ferramenta mais eficaz e menos custosa.

Palavras-chave: Aprendizado do computador. Inteligência artificial. Morango. Redes neurais (Computação).

ABSTRACT

In recent years, Machine Learning and Deep Learning techniques have been applied to strawberry research. Currently, identifying the maturity of strawberries efficiently and accurately is a challenge due to traditional methods, which are based on the appearance or chemical composition of the fruit, being expensive and time-consuming. Automatic classification of strawberries can offer farmers a more precise way to assess the quality of the fruits. This study proposes analyzing the efficiency of different combinations of supervised models and Convolutional Neural Networks (CNNs) to classify strawberry maturity. Seventy-one CNNs were tested for image feature extraction, followed by dimensionality reduction with PCA and the application of ten classifiers. The best results were obtained with CNNs from the ConvNeXt family (ConvNeXtBase, ConvNeXtSmall, and ConvNeXtTiny) and VGG (VGG16 and VGG19) in combination with Gradient Boosting, Histogram Based Gradient Boosting and SVM classifiers, achieving accuracy above 72% and F1-Score above 78%. The tested combinations proved efficient, providing a viable solution for the accurate classification of strawberry maturity, potentially benefiting farmers with a more effective and less costly tool.

Keywords: Artificial intelligence. Machine learning. Neural networks (Computer science). Strawberries.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE SIGLAS

CNN	*Convolutional Neural Network*
FLOPS	*Floating Point Operations per Second*
GAP	G*lobal Average Pooling*
GPU	*Graphics Processing Unit*
ILSVRC	*ImageNet Large Scale Visual Recognition Challenge*
k-FCV	*k-Fold Cross Validation*
LDA	*Linear Discriminant Analysis*
NASNet	*Neural Architecture Search Network*
PCA	*Principle Components Analysis*
QDA	*Quadratic Discriminant Analysis*
ReLU	*Rectified Linear Unit*
ResNet	*Residual Network*
RGB	*Red, Green, Blue*
RNN	*Recurrent Neural Network*
ROC	*Receiver Operating Characteristic*
SVM	*Support Vector Machines*
KNN	*K-Nearest Neighbors*
TanH	Tangente Hiperbólica
TPU	*Tensor Processing Units*
VGG	*Virtual Geometry Group*

SUMÁRIO

1 INTRODUÇÃO

O consumo de alimentos saudáveis e de forma equilibrada é essencial para ter uma boa saúde. É uma prática que ajuda no bem-estar físico, mental e social das pessoas (BEVILACQUA, 2006). Com o tempo, os consumidores brasileiros vêm valorizando cada vez mais os produtos relacionados à qualidade de vida (VIEIRA, 2010). Um dos alimentos preferidos pelos consumidores é o morango, por conta de suas características como aroma, textura suculenta e doçura. A maturação do morango tem sido avaliada, convencionalmente, por especialistas ou pesquisadores e são verificados aspectos como cor, textura e composição química para determinar a qualidade da fruta. Apesar de ter uma precisão satisfatória, esses métodos geralmente são destrutivos, demorados e trabalhosos. Por conta disso, criar um método rápido e não destrutivo para avaliar a maturação de morangos se torna uma abordagem interessante a se trabalhar (GAO *et al.*, 2020). Para esse problema é possível utilizar algoritmos de aprendizado de máquina que possam identificar automaticamente a qualidade dos morangos para serem colhidos.

O aprendizado de máquina e suas técnicas têm evoluído significativamente ao longo das últimas décadas, transformando diversas áreas tecnológicas e científicas. O aprendizado de máquina é uma subárea da inteligência artificial que se concentra no desenvolvimento de algoritmos que permitem aos computadores aprenderem a partir de dados (SHETTY *et al*., 2020). O aprendizado de máquina pode ser utilizado para realizar várias atividades que antes eram realizadas somente por seres humanos (TEIXEIRA, 2009). Uma dessas atividades é a classificação de dados, que é o agrupamento de objetos em determinadas classes com base em seus valores, e é amplamente utilizada na tomada de decisões (TEMBUSAI; MAWENGKANG; ZARLIS, 2021).

No início da história do aprendizado de máquina, foi introduzido o Perceptron por Rosenblatt (1958). O Perceptron é um modelo de rede neural simples que realiza classificações binárias. Com o tempo, surgiram algoritmos mais complexos, como as redes neurais multicamadas, que utilizam múltiplas camadas de Perceptrons para melhorar a capacidade de aprendizado dos modelos. A introdução do algoritmo de retropropagação, na década de 1980, foi um marco significativo, permitindo que as redes neurais ajustassem seus pesos com mais eficiência (SHETTY *et al*., 2020).

Redes neurais são estruturas computacionais inspiradas no cérebro humano, compostas por camadas de neurônios artificiais. Essas redes são capazes de aprender padrões complexos a partir dos dados, sendo amplamente utilizadas em diversas aplicações, como reconhecimento

de voz e imagens. Existem diferentes tipos de redes neurais, incluindo redes neurais *feedforward*, redes neurais recorrentes (RNNs) e redes neurais convolucionais (CNNs). As redes *feedforward* são as mais simples, com os dados fluindo em uma única direção, enquanto as RNNs são projetadas para lidar com dados sequenciais, como séries temporais e textos (SHETTY *et al*., 2020).

As CNNs são uma classe especial de redes neurais projetadas especificamente para processar e reconhecer padrões em imagens. Elas utilizam operações de convolução, onde filtros são aplicados sobre as imagens para extrair características importantes, como bordas e texturas. As CNNs foram popularizadas pelo trabalho de LeCun *et al.* (1999) e outros pesquisadores na década de 1990 e têm sido fundamentais em avanços recentes na visão computacional. As CNNs são amplamente utilizadas em tarefas como classificação de imagens, detecção de objetos e segmentação semântica (SHETTY *et al*., 2020).

O avanço das técnicas de aprendizado de máquina, redes neurais artificiais e redes neurais convolucionais tem revolucionado diversas áreas, permitindo a automação de tarefas que antes eram exclusivas de seres humanos. A utilização dessas tecnologias para a classificação de alimentos pode proporcionar um método mais eficiente e preciso para determinar a maturação dos morangos, garantindo melhor qualidade e redução de desperdícios. Futuramente, este estudo facilitará o desenvolvimento de aplicações inteligentes para este fim.

1.1 Objetivos

Este trabalho tem como objetivo geral facilitar aos agricultores a classificação precisa da aparência dos morangos em situações reais. Para isso, o objetivo específico deste projeto é analisar a eficiência de diferentes combinações de modelos de rede neural convolucional como extratores de características e de classificadores supervisionados, visando identificar as mais adequadas para a tarefa de classificação de morangos a partir de imagens. Assim, este trabalho poderia contribuir com os resultados de vários testes realizados como *benchmark*.

1.2 Organização do trabalho

Este trabalho está organizado em 6 capítulos, sendo o restante deste trabalho estruturado da seguinte forma. O Capítulo 2 traz a fundamentação teórica sobre aprendizado de máquina, aprendizado supervisionado, conceitos de alguns algoritmos desse tipo de aprendizado, redução de dimensionalidade, validação cruzada e métricas de avaliação de desempenho de algoritmos.

O Capítulo 3 apresenta sobre morangos, e depois alguns trabalhos relacionados ao tema desta pesquisa. No Capítulo 4 são descritas as metodologias utilizadas neste trabalho. O Capítulo 5 mostra os resultados obtidos durante este projeto de mestrado. E, finalmente, no Capítulo 6 são apresentadas as considerações finais deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos necessários para entender a metodologia proposta. A Seção 2.1 trata do aprendizado de máquina e uma de suas principais categorias. Na Seção 2.2 são mostrados alguns conceitos de algoritmos, como redes neurais artificiais, redes neurais convolucionais, e os modelos supervisionados. Na Seção 2.3 aborda a redução de dimensionalidade. A Seção 2.4 apresenta a validação cruzada. Finalmente, a Seção 2.5 trata sobre as métricas de avaliação de desempenho de algoritmos.

2.1 Aprendizado de máquina

O aprendizado de máquina é uma subárea dos estudos de Inteligência Artificial, que pesquisa o desenvolvimento de algoritmos que possibilitam a aprendizagem de novos conhecimentos, a adaptação com novas habilidades e formas de organizar o conhecimento já existente, e a busca da melhora de desempenho de forma automática por meio de experiências semelhantes aos que acontecem com os seres humanos (BLUM, 2007; MITCHELL, 1997).

Por conta do surgimento do aprendizado de máquina, os computadores podem realizar várias atividades que somente os humanos realizavam antes (TEIXEIRA, 2009). O objetivo principal, portanto, é aprender de forma automática a realizar o reconhecimento de padrões e a tomada de decisões (MITCHELL, 1997).

O aprendizado de máquina é utilizado quando o conhecimento humano não pode ser codificado em algoritmos, como no reconhecimento de objetos, uma tarefa complexa para descrever o processo decisório do cérebro (ALPAYDIN, 2010). A inteligência artificial e o reconhecimento de padrões são alguns campos que estão relacionados com o aprendizado de máquina (ALPAYDIN, 2010; MITCHELL, 1997).

O aprendizado supervisionado é uma categoria de aprendizado de máquina que utiliza conjuntos de dados que são rotulados com base no conhecimento de especialistas de uma determinada área (HAYKIN, 1999). Esse aprendizado infere uma função ou regra para poder atribuir uma classe a um objeto nunca visto antes a partir de dados de treinamento previamente classificados (ALPAYDIN, 2010; MITCHELL, 1997). Ou seja, o objetivo do aprendizado supervisionado é prever uma saída para cada dado de entrada, baseando-se no padrão aprendido em cada uma das classes no treinamento realizado com dados rotulados (MITCHELL, 1997).

A seguir, serão apresentados os principais conceitos de alguns dos algoritmos de aprendizado supervisionado.

2.2 Algoritmos

Nesta seção serão apresentados os conceitos de SVM, regressão logística, KNN, Naive Bayes, árvore de decisão, *Random Forest*, redes neurais artificiais e redes neurais convolucionais, com algumas de suas arquiteturas.

2.2.1 SVM

Support Vector Machine (VAPNIK; LERNER, 1963) é uma técnica de aprendizado de máquina utilizada tanto para classificação quanto para regressão, sendo especialmente eficaz em problemas de alta dimensionalidade. Desenvolvidas no contexto da teoria de aprendizagem estatística, as SVMs buscam encontrar o hiperplano ótimo que separa os dados em diferentes classes com a maior margem possível. Durante o treinamento, a SVM resolve um problema de otimização para maximizar a distância entre o hiperplano e os pontos de dados mais próximos de cada classe, chamados de vetores de suporte. Esses vetores são cruciais, pois determinam a posição do hiperplano. Ao maximizar a margem entre o hiperplano e os vetores de suporte, a SVM assegura que o modelo tenha uma melhor capacidade de generalizar para novos dados não vistos (AWAD; KHANNA, 2015).

O SVM é eficaz em evitar o *overfitting*, e usando diferentes funções de *kernel*, o SVM pode ser adaptado para resolver problemas não-lineares mapeando os dados para espaços de maior dimensão onde se tornam linearmente separáveis (AWAD; KHANNA, 2015).

2.2.2 Regressão Logística

A regressão logística é um modelo estatístico usado para prever a probabilidade de ocorrência de um evento. Diferente da regressão linear, que prevê valores contínuos, a regressão logística é utilizada quando o resultado é binário, ou seja, tem duas possibilidades (AWAD; KHANNA, 2015).

A fórmula básica da regressão logística é:

$$P(Y|X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}, \quad (1)$$

onde P(Y|X) é a probabilidade de Y ocorrer dado X, e é a base do logaritmo natural, β_0 é o intercepto e β_1 é o coeficiente que multiplica a variável X. A função logística transforma a

combinação linear dos parâmetros β_0 e $\beta_1 \cdot X$ em uma probabilidade entre 0 e 1. Os coeficientes β_0 e β_1 são estimados usando uma técnica chamada de máxima verossimilhança. Basicamente, se encontram os valores desses coeficientes que tornam os dados observados mais prováveis (AWAD; KHANNA, 2015).

2.2.3 KNN

O algoritmo *K-Nearest Neighbors* (KNN) é uma técnica de classificação que atribui uma nova amostra à classe predominante entre seus k vizinhos mais próximos. Esse método é baseado na premissa de que objetos próximos no espaço de características tendem a pertencer à mesma classe (BISHOP, 2006).

Para determinar a classe de uma nova amostra, o algoritmo segue os seguintes passos: escolhe-se um valor para k, que representa o número de vizinhos mais próximos a serem considerados; calculam-se as distâncias entre a nova amostra e todas as amostras do conjunto de treinamento com a distância Euclidiana ou outras métricas de distância; identificam-se os K vizinhos mais próximos da nova amostra com base nas distâncias calculadas; e a nova amostra é classificada na classe que tiver o maior número de representantes entre os K vizinhos. Em caso de empate, a decisão pode ser feita aleatoriamente ou com base em critérios adicionais (BISHOP, 2006).

O valor de k tem um impacto significativo no desempenho do algoritmo. Valores menores de K tendem a produzir modelos mais sensíveis às peculiaridades do conjunto de dados de treinamento, resultando em uma fronteira de decisão complexa e possivelmente irregular. Por outro lado, valores maiores de k produzem modelos mais suaves, que podem generalizar melhor, mas podem também perder detalhes importantes das classes (BISHOP, 2006).

Uma propriedade interessante do classificador do vizinho mais próximo ($k = 1$) é que, no limite onde o número de amostras N tende ao infinito, a taxa de erro não será mais do que o dobro da taxa de erro mínima alcançável por um classificador ideal que utiliza as verdadeiras distribuições de classes (BISHOP, 2006).

Embora seja um método intuitivo e fácil de implementar, o KNN possui algumas limitações. A principal delas é a necessidade de armazenar todo o conjunto de dados de treinamento, o que pode ser computacionalmente custoso para conjuntos de dados muito grandes. Além disso, a classificação de novas amostras pode ser lenta, pois envolve o cálculo das distâncias para todas as amostras do conjunto de treinamento (BISHOP, 2006).

2.2.4 Naive Bayes

O modelo de Naive Bayes é um método probabilístico simples utilizado para classificação, que se baseia na aplicação do teorema de Bayes com uma suposição forte de independência entre os atributos. Isso significa que a presença de um atributo em uma classe é considerada independente da presença de qualquer outro atributo. Este modelo é particularmente eficaz para grandes volumes de dados e é amplamente utilizado em diversas áreas, incluindo a classificação de texto e a filtragem de *spam* (AWAD; KHANNA, 2015; BISHOP, 2006).

Para ajustar o modelo aos dados de treinamento rotulados, utiliza-se a máxima verossimilhança, assumindo que os dados são gerados de forma independente do modelo. O ajuste é realizado separadamente para cada classe, utilizando os dados rotulados correspondentes. Por exemplo, se a densidade de probabilidade dentro de cada classe for escolhida como Gaussiana, a suposição de Naive Bayes implica que a matriz de covariância de cada Gaussiana é diagonal, resultando em contornos de densidade constantes alinhados com os eixos. Esta suposição simplificadora é útil especialmente quando a dimensionalidade do espaço de entrada é alta, pois a estimação de densidade no espaço completo de D dimensões seria mais desafiadora. Além disso, é vantajosa quando o vetor de entrada contém tanto variáveis discretas quanto contínuas, permitindo representações separadas para cada tipo de variável (BISHOP, 2006).

Apesar da suposição de independência condicional ser bastante forte e possivelmente levar a representações pobres das densidades condicionais da classe, o modelo de Naive Bayes pode ainda assim proporcionar um bom desempenho na classificação. Isso ocorre porque as fronteiras de decisão podem ser insensíveis a alguns detalhes nas densidades condicionais da classe (BISHOP, 2006).

2.2.5 Árvore de Decisão

As árvores de decisão são uma ferramenta amplamente utilizada em análise de dados e aprendizado de máquina. Elas fornecem uma maneira intuitiva e visual de tomar decisões baseadas em dados, facilitando a compreensão e a comunicação dos resultados. Em termos simples, esse algoritmo ajuda a tomar decisões com base em uma série de perguntas. Uma árvore de decisão é composta por três partes principais: nós de decisão, que representam as perguntas ou testes sobre os dados; ramos, que representam as respostas possíveis para cada

pergunta; e folhas, que representam a decisão final ou a classificação (ROKACH; MAIMON, 2005).

Para determinar a estrutura da árvore de decisão, mesmo para um número fixo de nós na árvore, o problema de determinar a estrutura ideal (incluindo a escolha da variável de entrada para cada divisão, bem como o valor do limiar correspondente) é geralmente computacionalmente inviável devido ao grande número de soluções possíveis. Em vez disso, uma otimização gananciosa é geralmente feita começando com um único nó raiz, correspondendo a todo o espaço de entrada, e depois expandindo a árvore adicionando nós um de cada vez. A cada etapa, haverá um número de regiões candidatas no espaço de entrada que podem ser divididas, correspondendo à adição de um par de nós folha à árvore existente. Para cada uma dessas regiões, há a escolha de qual das variáveis de entrada dividir, bem como o valor do limiar. A otimização conjunta da escolha da região a ser dividida, e a escolha da variável de entrada e do limiar, pode ser feita de forma eficiente por meio de uma busca exaustiva, observando que, para uma dada escolha de variável de divisão e limiar, a escolha ótima da variável preditiva é dada pela média local dos dados (BISHOP, 2006).

Uma vez que a estratégia gananciosa para expandir a árvore é estabelecida, permanece a questão de quando parar de adicionar nós. Uma abordagem simples seria parar quando a redução no erro residual cair abaixo de algum limiar. No entanto, empiricamente, verifica-se que muitas vezes nenhuma das divisões disponíveis produz uma redução significativa no erro, e ainda assim, após várias outras divisões, uma redução substancial no erro é encontrada. Por essa razão, é prática comum crescer uma árvore grande, usando um critério de parada baseado no número de pontos de dados associados aos nós folha, e depois podar a árvore resultante. A poda é baseada em um critério que equilibra o erro residual contra uma medida de complexidade do modelo (BISHOP, 2006).

Algumas das vantagens das árvores de decisão são que como elas se parecem com um fluxograma, as pessoas sem mesmo ter formação técnica podem entendê-las, podem lidar com diferentes tipos de dados, tanto numéricos quanto categóricos e não fazem suposições sobre a distribuição dos dados, o que as torna versáteis para diversos tipos de problemas (ROKACH; MAIMON, 2005).

Apesar das vantagens, as árvores de decisão estão propensas a *overfitting*, elas podem se ajustar muito bem aos dados de treinamento, perdendo a capacidade de generalizar para novos dados. Além disso, árvores muito grandes e complexas podem se tornar difíceis de interpretar, e são sensíveis a variações nos dados, incluindo *outliers* e ruído (ROKACH; MAIMON, 2005).

2.2.6 *Gradient Boosting*

Gradient Boosting é uma técnica usada em aprendizado de máquina para construir modelos de previsão. A ideia principal é combinar vários modelos simples de árvores de decisão para formar um modelo mais robusto. Esse método se destaca por sua capacidade de melhorar a precisão preditiva ao minimizar os erros cometidos pelos modelos individuais (FRIEDMAN, 2001).

No *Gradient Boosting*, os modelos são construídos um após o outro. Cada novo modelo corrige os erros cometidos pelos modelos anteriores. Quando um modelo é criado, ele avalia onde errou. Em seguida, um novo modelo é treinado para corrigir especificamente esses erros. Esse processo se repete várias vezes, com cada modelo novo focando nas áreas onde os anteriores falharam. Depois de criar vários modelos, todos são combinados para fazer a previsão final. A combinação é feita de forma a dar mais peso aos modelos que foram mais precisos, resultando em uma previsão geral muito mais robusta (FRIEDMAN, 2001).

Devido ao foco contínuo na correção de erros, o *Gradient Boosting* costuma produzir previsões muito precisas. Esse método pode ser aplicado a diversos tipos de dados e problemas, desde previsões de vendas até diagnósticos médicos. O *Gradient Boosting* é particularmente eficaz em lidar com dados complexos, onde a relação entre as variáveis pode não ser evidente (FRIEDMAN, 2001).

2.2.7 *Histogram-based Gradient Boosting*

O *Histogram-Based Gradient Boosting* é uma variação de *Gradient Boosting* que melhora a eficiência do processo de treinamento. Em vez de considerar todos os valores possíveis dos dados para decidir onde dividir, essa técnica agrupa os dados em histogramas, que são basicamente contagens de frequências de intervalos de valores. Isso reduz significativamente o número de divisões possíveis, o que acelera o processo de treinamento (FRIEDMAN, 2001).

Os histogramas são utilizados para criar *bins*, ou intervalos, que contêm dados semelhantes. Durante o processo de treinamento, o algoritmo verifica apenas esses *bins* em vez de todos os valores individuais dos dados. Isso reduz o tempo computacional necessário para encontrar as divisões ótimas, sem sacrificar a precisão do modelo (FRIEDMAN, 2001).

2.2.8 *Random Forest*

Uma *Random Forest* é uma técnica de aprendizado de máquina que combina muitas árvores de decisão para melhorar a precisão das previsões. Em vez de confiar em uma única árvore de decisão complexa, uma *Random Forest* usa muitas árvores menores e mais simples. Cada uma dessas árvores menores faz sua previsão com base em um subconjunto aleatório dos dados. Ao combinar as previsões de todas essas árvores, a *Random Forest* pode fazer uma previsão mais precisa e estável do que qualquer árvore individual (AWAD; KHANNA, 2015).

O algoritmo de *Random Forest* começa com a criação de árvores, pegando várias amostras aleatórias do conjunto de dados original. Para cada amostra, uma árvore de decisão é gerada. Em cada ponto onde a árvore se divide (nós), uma seleção aleatória de variáveis (características) é escolhida para decidir a melhor forma de dividir os dados. Quando um novo ponto de dados precisa ser classificado, cada árvore na floresta dá uma previsão, e a previsão mais comum é escolhida (AWAD; KHANNA, 2015).

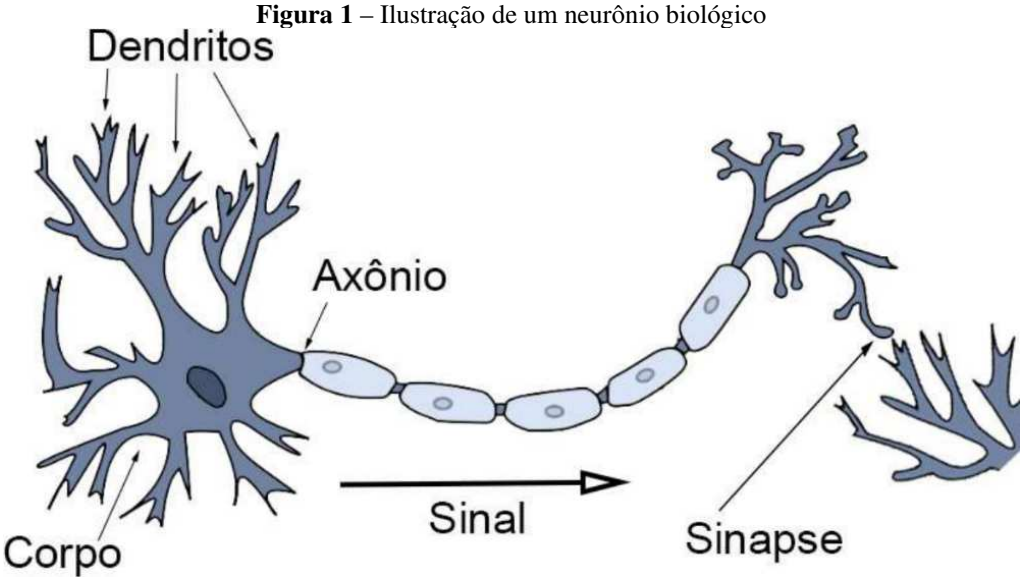
As *Random Forests* são boas para lidar com grandes conjuntos de dados e podem funcionar bem mesmo quando faltam alguns dados. Elas também ajudam a determinar quais variáveis são mais importantes para fazer previsões precisas. Ao fazer a média de várias árvores, as previsões são mais precisas e, além disso, podem identificar quais variáveis são mais importantes para fazer previsões (AWAD; KHANNA, 2015).

2.2.9 Redes neurais artificiais

As redes neurais artificiais, em geral, são máquinas que utilizam neurônios artificiais, baseados nos neurônios biológicos, que imitam o cérebro humano na realização de uma tarefa específica, onde se aplica um processo de aprendizagem. O cérebro humano é capaz de organizar os neurônios para processar informações de forma paralela e, dependendo da atividade, é mais rápido do que as máquinas existentes até o momento. Alguns exemplos de processamento realizado por redes neurais são o reconhecimento de padrões, percepção e coordenação motora (HAYKIN, 1999).

No neurônio biológico, geralmente os sinais são recebidos de outros neurônios pelos dendritos e as conexões são feitas pelas sinapses. Quando recebe um estímulo são definidos os sinais de entrada, depois são reunidos no corpo de um neurônio para gerar um sinal de saída para ser transmitido pelo axônio e, assim, para outros neurônios (SÜDHOF, 2021). A Figura 1 mostra uma imagem das partes principais de um neurônio biológico.

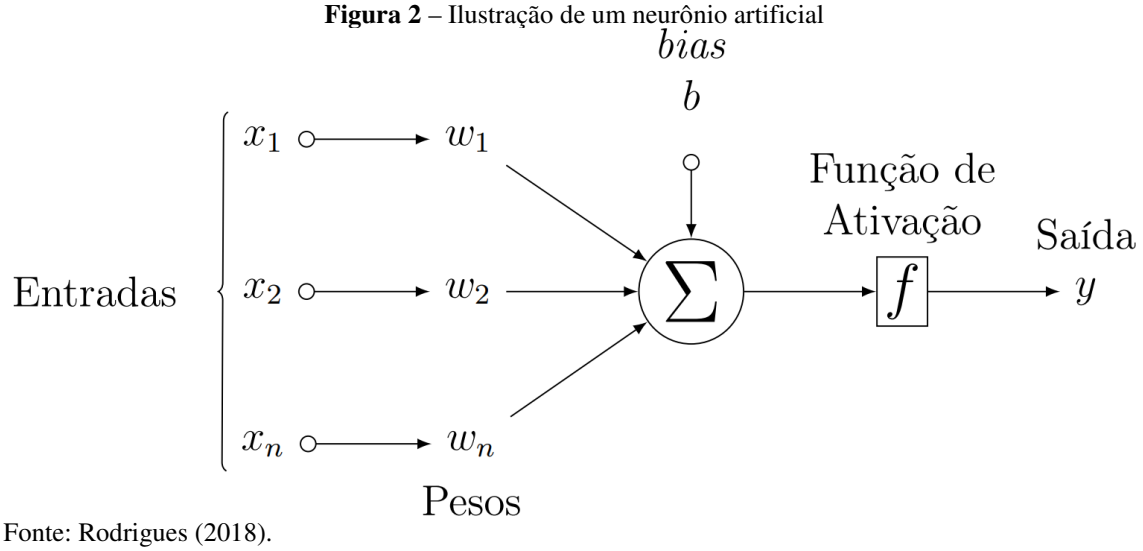
Figura 1 – Ilustração de um neurônio biológico



Fonte: Rodrigues (2018).

Basicamente, existem os estímulos de entrada, as ligações sinápticas e as saídas nos neurônios artificiais. Porém, os neurônios artificiais não são tão complexos como os biológicos. Por esse motivo existem várias pesquisas relacionadas às redes neurais artificiais (HAYKIN, 1999). A Figura 2 ilustra o modelo de um neurônio artificial.

Figura 2 – Ilustração de um neurônio artificial



Fonte: Rodrigues (2018).

A Equação 1 mostra a modelagem matemática de um neurônio artificial (RODRIGUES, 2018):

$$y_k = f\left[\left(\sum_{i=1}^{n} x_i * w_{ki}\right) + b_k\right] \quad (2)$$

em que a saída do neurônio k é representada por y_k. É realizada uma somatória de cada entrada x_i multiplicada pelos respectivos pesos w_i de cada neurônio k, gerando um produto interno, e tudo isso é somado ao bias b do neurônio k, que aumenta ou diminui a entrada da função de ativação f (RODRIGUES, 2018). A função de ativação é uma função algébrica que decide se um neurônio será ativado ou não (HAYKIN, 1999), limitando a amplitude do sinal de saída de um neurônio (HAYKIN, 1999; RUSSELL; NORVIG, 2009). A ativação de um neurônio depende do valor do produto interno gerado ser maior ou menor do que um determinado limiar de ativação, o qual é definido dependendo da função de ativação escolhida. Caso o valor seja maior, o neurônio será ativado (RUSSELL; NORVIG, 2009). A seguir são definidas brevemente algumas das funções de ativação mais conhecidas: identidade ou linear; sigmóide; tangente hiperbólica; e *Rectified Linear Unit*.

A função de ativação identidade ou linear é uma função simples que executa a operação de identidade nos dados, considerando que existem uma proporção entre os dados de entrada e os dados de saída. Geralmente é utilizada em uma rede neural artificial para neurônios na camada de entrada. A definição da função é (RODRIGUES, 2018):

$$f(x) = x. \tag{3}$$

A função sigmóide é parecida com o funcionamento dos neurônios biológicos, em que se decide se um neurônio será ativado ou não com base nas saídas entre 0 e 1, motivo pelo qual essa função foi muito utilizada em redes neurais artificiais (RODRIGUES, 2018). Porém, a função sigmóide não é sensível às mudanças pequenas quando o argumento for extremamente positivo ou negativo, gerando dificuldades no treinamento, como a instabilidade que surge durante a classificação. A função sigmoide é dada por (RODRIGUES, 2018):

$$\sigma(x) = \frac{1}{1 + e^x}. \tag{4}$$

A função tangente hiperbólica (TanH) é parecida com a função identidade ou linear e é uma alternativa para poder ativar camadas ocultas das redes neurais artificiais ao invés de utilizar a sigmoide. A função de ativação é dada por (RODRIGUES, 2018):

$$TanH(x) = 2\sigma(2x) - 1. \tag{5}$$

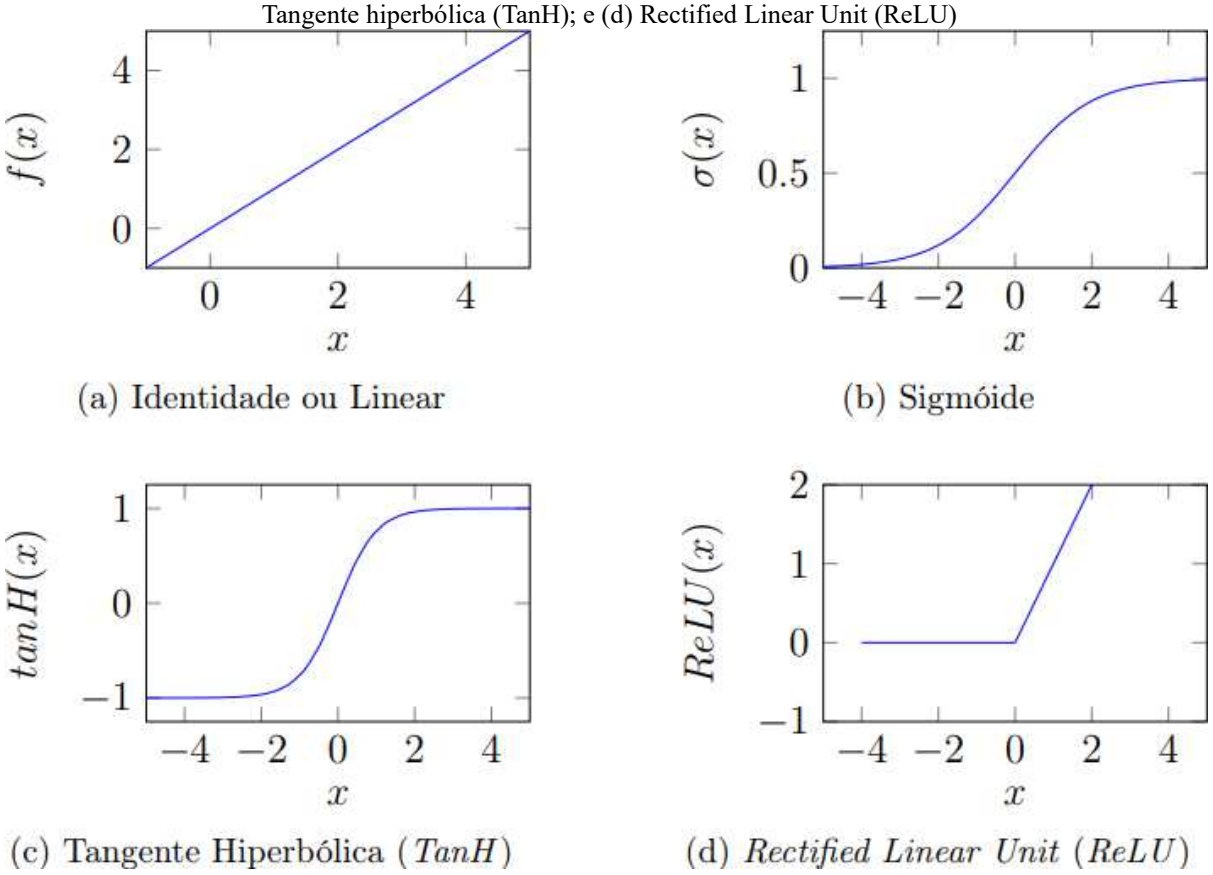
E a função *Rectified Linear Unit* (ReLU) permite aprender modelos não lineares (HAKIM; FADHIL, 2021) e tem como a função definida por (RODRIGUES, 2018):

$$ReLU(x) = \max\{0, x\}, \tag{6}$$

sendo que se a entrada for menor do que 0, a saída é igual a 0 e, se a entrada for maior do que 0, a saída é igual à entrada (RODRIGUES, 2018).

A Figura 3 mostra graficamente as funções apresentadas.

Figura 3 – Exemplos de funções de ativação mais conhecidas: (a) Identidade ou Linear; (b) Sigmóide; (c) Tangente hiperbólica (TanH); e (d) Rectified Linear Unit (ReLU)

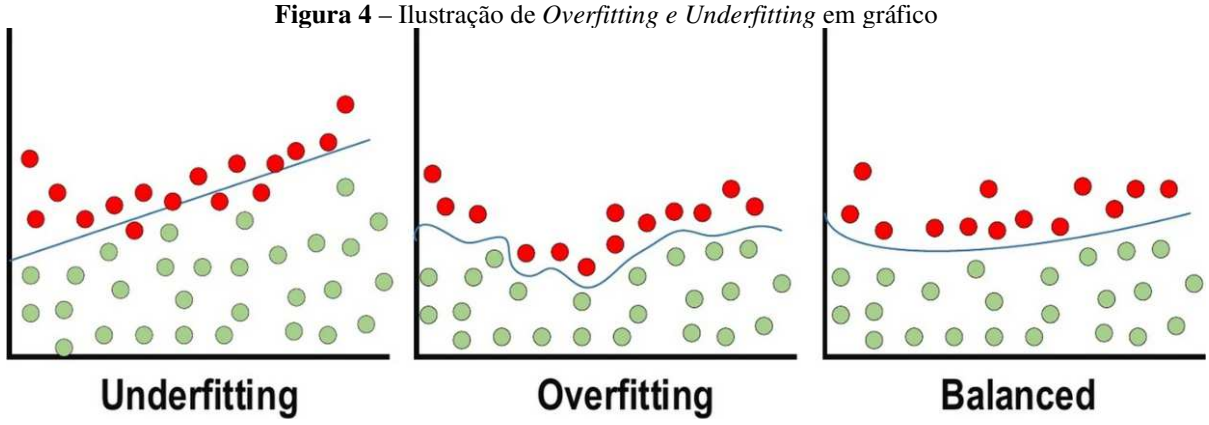


Fonte: Rodrigues (2018).

As redes neurais artificiais são métodos que podem ser aplicados no contexto geral, como a interpretação de dados complexos, para aprender funções reais por meio de treinamentos com dados de entrada (MITCHELL, 1997).

A realização de um treinamento para obter um resultado satisfatório depende da arquitetura da rede a ser escolhida e do número de iterações para que não ocorra dois problemas muito comuns, o *overfitting* e o *underfitting*. O *overfitting* é a falta de generalização por conta do treinamento excessivo, o que significa que tem um número de acertos grande no conjunto de treinamento, mas tem poucos acertos no conjunto de teste. E o *underfitting* ocorre quando não consegue obter um número de acertos considerável no conjunto de treinamento pelo modelo ter sido pouco treinado (GOODFELLOW; BENGIO; COURVILLE, 2016). A Figura 4 ilustra o *overfitting* e o *underfitting*, assim como um resultado de um modelo treinado adequadamente.

Figura 4 – Ilustração de *Overfitting e Underfitting* em gráfico



Fonte: Alzubaidi *et al.* (2021).

Dentro do aprendizado de máquina, um dos algoritmos que tem mais destaque é o Perceptron. O algoritmo Perceptron, introduzido por Frank Rosenblatt (1958), é um modelo de discriminação linear essencial na história do reconhecimento de padrões. Este modelo é utilizado para classificar dados em duas classes distintas (AHMAD *et al*., 2021). A entrada do Perceptron é um vetor de características $\boldsymbol{x}$, que é transformado através de uma função não-linear fixa para gerar um vetor de características $\boldsymbol{\phi(x)}$ multiplicado pelo peso w^T. O modelo linear generalizado resultante é expresso pela função (ROSENBLATT, 1958):

$$y(\boldsymbol{x}) = f(\, w^T \phi(\boldsymbol{x})) \tag{7}$$

onde a função de ativação $f(\mathrm{a})$ é uma função degrau definida como:

$$\begin{cases} +1 \; se \; a \geq 0 \\ -1 \; se \; a < 0 \end{cases}$$

Ele funciona aprendendo a separar os dados com uma linha reta (ou hiperplano em dimensões mais altas). Cada vez que o algoritmo faz uma previsão errada, ele ajusta sua linha de separação para melhorar as próximas previsões. O algoritmo Perceptron começa com uma linha de separação arbitrária. Para cada ponto de dados, faz uma previsão de qual lado da linha o ponto está. Se a previsão estiver errada, a linha de separação é ajustada somando ou subtraindo o ponto de erro da linha atual para corrigir o erro (AHMAD *et al*., 2021).

O teorema da convergência do Perceptron garante que, se existe uma solução exata (ou seja, se os dados de treinamento são linearmente separáveis), o algoritmo de aprendizado do Perceptron encontrará essa solução em um número finito de passos. No entanto, o número de passos para a convergência pode ser substancial. Além disso, para conjuntos de dados que não são linearmente separáveis, o algoritmo nunca convergirá (ROSENBLATT, 1958).

Em situações reais, às vezes os dados não são apresentados de forma perfeita. O algoritmo Perceptron pode falhar nesses casos, porque ele não consegue distinguir entre os dados manipulados e os dados reais. No entanto, foram desenvolvidas versões aprimoradas do

Perceptron para lidar com essas situações, fazendo ajustes que consideram possíveis manipulações nos dados (AHMAD *et al*., 2021).

A rede neural artificial faz parte de uma área chamada de *Deep learning* (aprendizado profundo). *Deep Learning* é uma subárea de *Machine Learning* e é baseado nos padrões de como o cérebro humano processa as informações. *Deep Learning* é uma forma de explorar várias camadas de processamento de informações não-lineares por meio de técnicas de *Machine Learning*, de maneira que os algoritmos consigam aprender múltiplos níveis de representação e abstração, dando sentido aos dados (DENG; YU, 2014). A arquitetura dos algoritmos de aprendizado profundo é formada por multicamadas, em que as características de baixo nível são extraídas nas primeiras camadas e as características de alto nível são extraídas nas camadas mais profundas. Essa extração é feita de forma automática e na saída do processo realiza-se uma classificação dos objetos encontrados, sendo considerado o principal benefício do aprendizado profundo (ALZUBAIDI *et al.*, 2021).

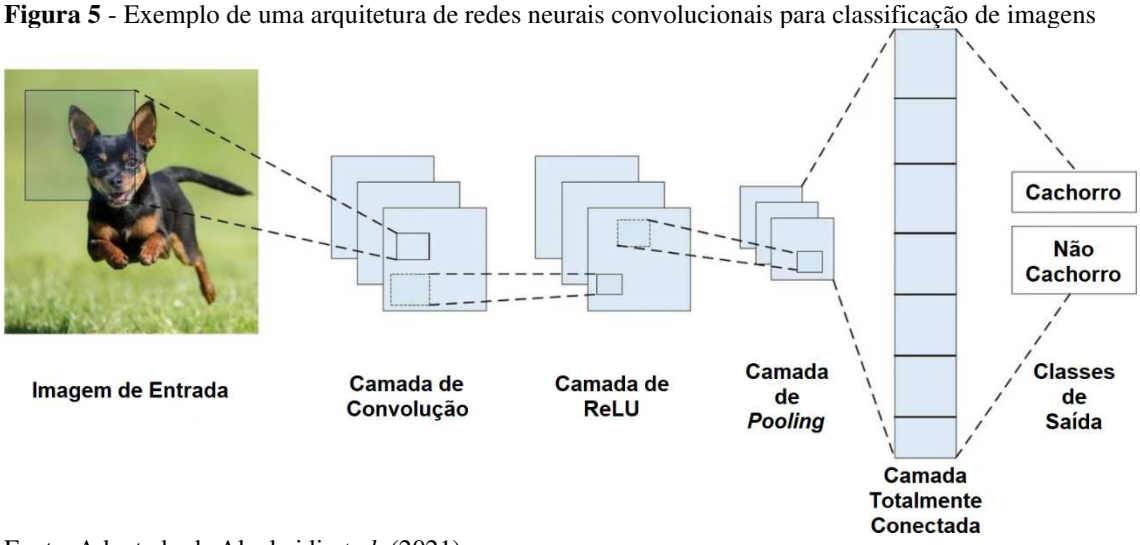
O aprendizado de máquina depende muito da integridade e de como estão representados os dados de entrada para obter um bom desempenho nos modelos. Assim, existem várias pesquisas que têm como foco a extração de características para poder gerar essas características por meio de dados brutos, e para isso geralmente precisa de muito esforço dos pesquisadores (ALZUBAIDI *et al.*, 2021).

Dentro do contexto do aprendizado profundo pode-se destacar as redes neurais convolucionais (ALZUBAIDI *et al.*, 2021), que serão discutidas a seguir.

2.2.10 Redes neurais convolucionais

As redes neurais convolucionais, do inglês *Convolutional Neural Network* (CNN), propostas inicialmente por Fukushima (1979), têm sido muito eficientes no reconhecimento de imagens, incentivando o seu uso em várias áreas de pesquisa. Uma CNN é formada basicamente por camadas convolucionais, *pooling* e camadas totalmente conectadas para extrair as características dos dados de forma automática (ALZUBAIDI *et al.*, 2021; RODRIGUES, 2018). A Figura 5 mostra um exemplo de um modelo de uma CNN.

Figura 5 - Exemplo de uma arquitetura de redes neurais convolucionais para classificação de imagens



Fonte: Adaptado de Alzubaidi *et al.* (2021).

Em um modelo CNN, a entrada de cada camada tem altura, largura e profundidade, ou seja, tem três dimensões, em que geralmente a altura da imagem a ser utilizada equivale à largura e a profundidade é definida pelo número de canais, como por exemplo o canal único para escala de cinza de uma imagem, com a profundidade igual a um, e os canais RGB de uma imagem, com a profundidade igual a três. Os *kernels* (filtros) são fundamentais para realizar as conexões locais e os parâmetros dos *kernels* estão associados com os parâmetros de *bias* e pesos, criando mapas de características. Existem vários *kernels* em cada camada convolucional. Cada *kernel* possui um certo número de dimensões (no caso de imagens coloridas RGB possui 3 dimensões) e cada um equivale a vários receptores distintos que respondem para cada tipo de característica (HAKIM; FADHIL, 2021). Em geral, os valores dos pesos do *kernel* iniciam de forma aleatória no treinamento. A cada iteração esses pesos são ajustados, aprendendo a extrair recursos significativos (ALZUBAIDI *et al.*, 2021).

No processo de convolução, um filtro passa por todos os elementos existentes em uma imagem e cada um dos elementos é multiplicado pelo respectivo peso e depois somado para produzir um mapa de características (ALZUBAIDI *et al.*, 2021; HAKIM; FADHIL, 2021).

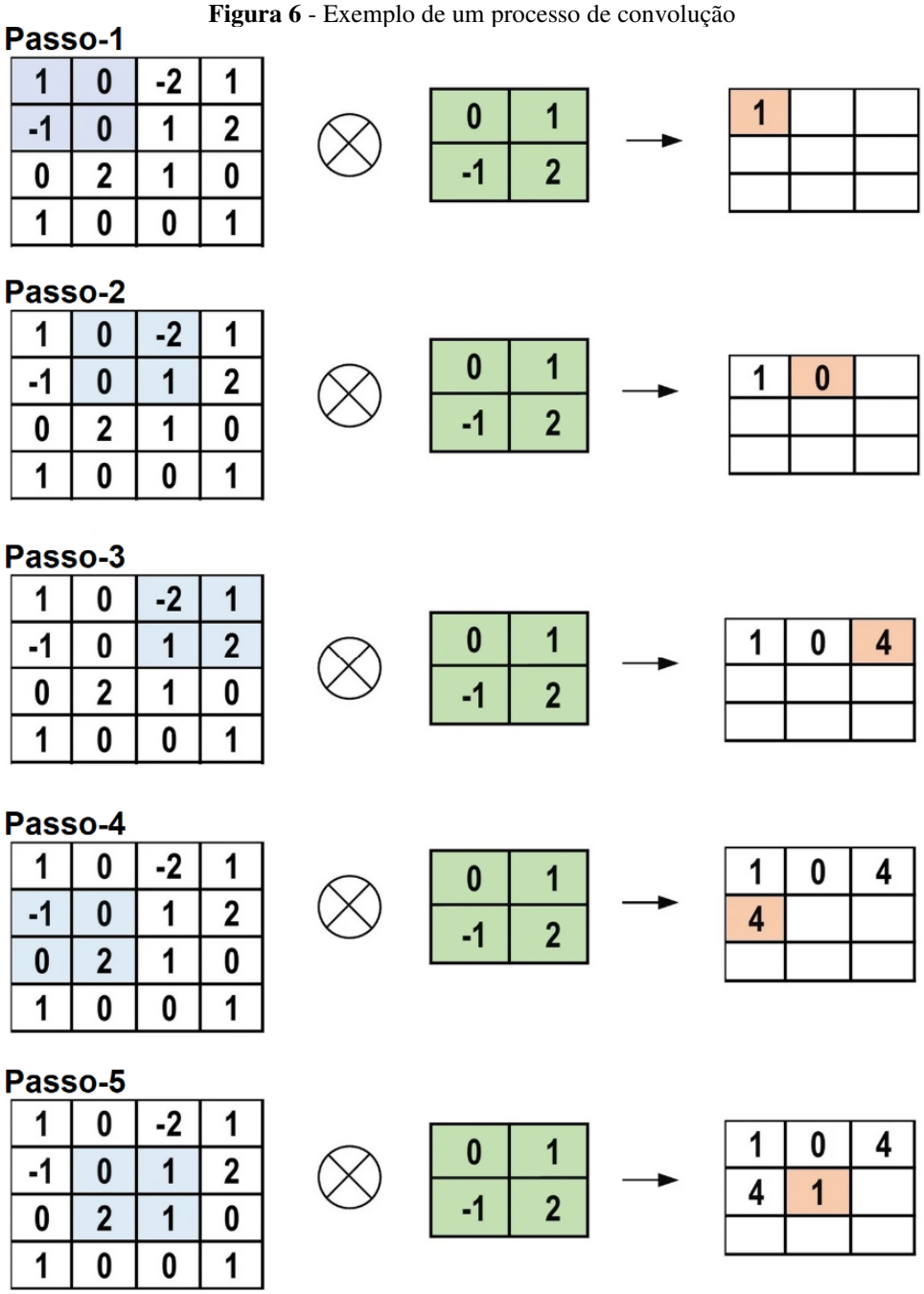
A convolução é definida por (HAKIM; FADHIL, 2021):

$$x_{i,j}^{l} = \sum_{a} \sum_{b} w_{a,b}^{(l-1,f)} \, y_{i+a,j+b}^{(l-1)} + bias^{f}, \tag{8}$$

onde $y_{i,j}^{l}$ é a saída da camada l e $w_{a,b}^{(l-1,f)}$ é o peso do *kernel* f que é aplicado na camada $l-1$.

A Figura 6 mostra um exemplo de processo de convolução sendo realizado em uma imagem 4x4, em escala de cinza, com um *kernel* 2x2, resultando no mapa de características (ALZUBAIDI *et al.*, 2021; HAKIM; FADHIL, 2021).

Figura 6 - Exemplo de um processo de convolução

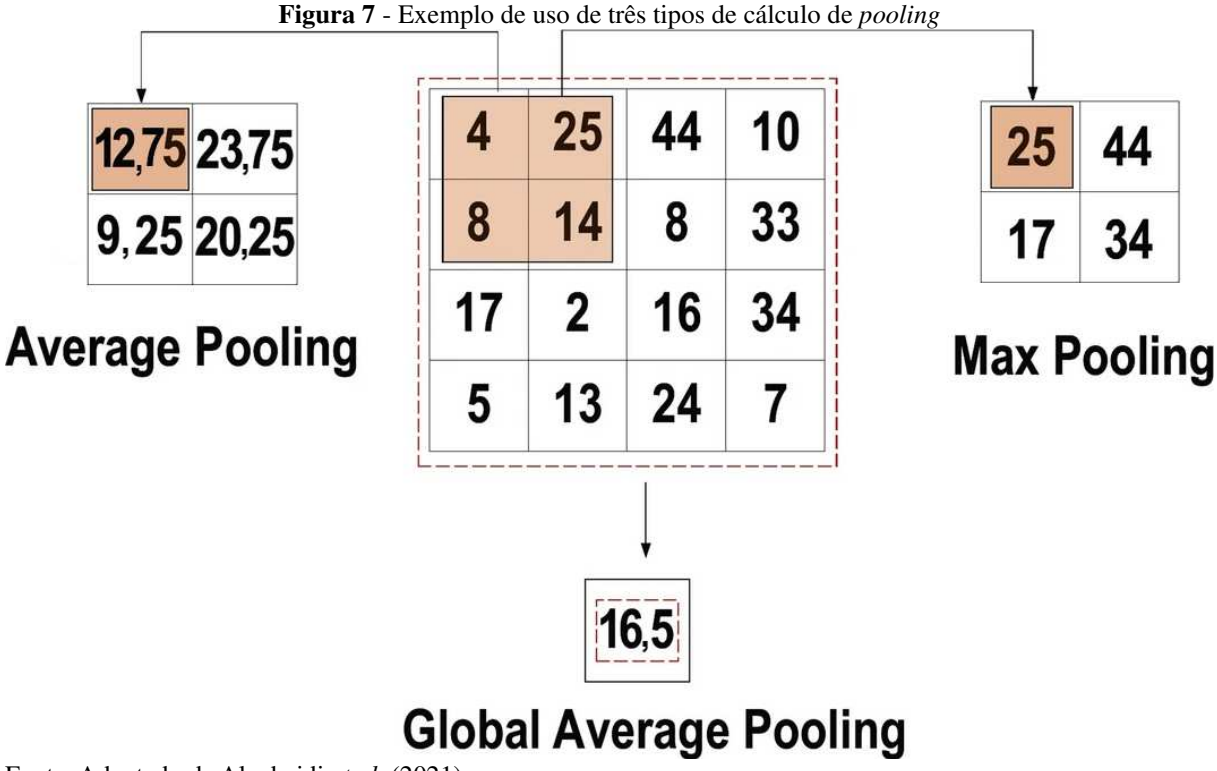


Fonte: Adaptado de Alzubaidi *et al.* (2021).

Após o processo de convolução, os mapas de características produzidos podem passar pela camada de *pooling*, que reduz a resolução desses mapas, ou seja, reduz a quantidade de atributos, mantendo as informações mais relevantes e gerando representações mais compactas. As unidades de um mapa de características combinam a entrada de um pequeno conjunto *n* x *n* de unidades, chamado de janela. Essa janela de *pooling* pode ter um tamanho qualquer e as janelas podem ser sobrepostas (SCHERER; MÜLLER; BEHNKE, 2010).

Para realizar essa redução existem vários tipos de *pooling*, sendo mais conhecidos e mais utilizados *max* (máximo), *min* (mínimo) e *average* (média) (GALANIS *et al.*, 2022). O *max-pooling* seleciona o maior valor do mapa de características dentro de uma determinada janela, e cada um desses valores selecionados é proporcional à probabilidade de ser uma classe de um objeto (ALZUBAIDI *et al.*, 2021). O *min-pooling* seleciona o menor valor do mapa de característica dentro de uma determinada janela (GALANIS *et al.*, 2022). O *average-pooling* funciona da mesma maneira que o *max-pooling*, mas ele seleciona a média entre os elementos da janela. Existe também o *global average pooling* (GAP), em que se calcula a média dos elementos do mapa de características como um todo (ALZUBAIDI *et al.*, 2021). A Figura 7 ilustra um exemplo da forma como o *max-pooling*, o *average-pooling* e o GAP selecionam os elementos de um mapa de características com uma janela 2x2 com passo 2.

Figura 7 - Exemplo de uso de três tipos de cálculo de *pooling*



Fonte: Adaptado de Alzubaidi *et al.* (2021).

E, finalmente, a saída da camada anterior (podendo ser a camada de convolução ou a de *pooling*, dependendo da arquitetura da rede neural) passa pela camada totalmente conectada (*Fully Connected*), onde cada neurônio se conecta com todos os neurônios da camada anterior, e a camada totalmente conectada realiza a classificação com base na saída da camada anterior (ALZUBAIDI *et al.*, 2021; HAKIM; FADHIL, 2021). Algumas das arquiteturas de redes neurais convolucionais são apresentadas a seguir.

2.2.11 VGG

As redes VGG16 e VGG19 são arquiteturas de redes neurais convolucionais desenvolvidas pelo *Visual Geometry Group* da Universidade de Oxford, proposta por Simonyan e Zisserman (2015). Essas arquiteturas são conhecidas por sua profundidade, utilizando um grande número de camadas convolucionais para melhorar a precisão em tarefas de reconhecimento de imagem em grande escala. A principal contribuição dessas arquiteturas é a utilização de filtros convolucionais muito pequenos, de 3x3 pixels, que permitem capturar a noção de esquerda/direita, cima/baixo e centro com eficiência (SIMONYAN; ZISSERMAN, 2015).

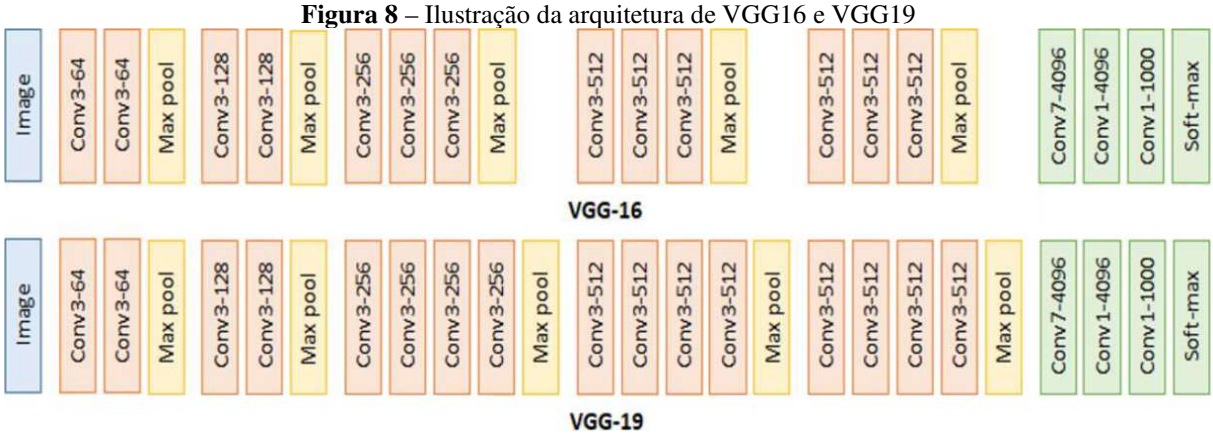
A arquitetura VGG16 consiste em 16 camadas com pesos treináveis. O design da rede começa com a entrada de uma imagem RGB de tamanho fixo de 224x224 pixels. O único pré-processamento aplicado é a subtração do valor médio dos pixels RGB calculado sobre o conjunto de treinamento. A imagem é então passada por uma pilha de camadas convolucionais, todas com filtros de 3x3 pixels e com *stride* de 1 pixel. O *padding* é ajustado para garantir que a resolução espacial seja preservada após cada convolução (SIMONYAN; ZISSERMAN, 2015).

As camadas convolucionais são seguidas por camadas de *pooling* máximo (*max-pooling*) realizadas sobre janelas de 2x2 pixels com *stride* de 2. Após várias camadas convolucionais e de *pooling*, a rede possui três camadas totalmente conectadas (Fully-Connected - FC): as duas primeiras com 4096 canais cada e a terceira com 1000 canais para classificação das 1000 categorias do ILSVRC. A última camada é uma camada *soft-max*. Todas as camadas ocultas utilizam a não-linearidade de retificação (ReLU) (SIMONYAN; ZISSERMAN, 2015).

A arquitetura VGG19 é uma versão mais profunda da VGG16, composta por 19 camadas com pesos treináveis. A configuração segue os mesmos princípios da VGG16, utilizando filtros convolucionais de 3x3 pixels e camadas de *pooling* máximo. A principal diferença reside na adição de mais camadas convolucionais, aumentando a capacidade da rede de aprender características mais complexas das imagens (SIMONYAN; ZISSERMAN, 2015).

A Figura 8 mostra a arquitetura da VGG16 e VGG19.

Figura 8 – Ilustração da arquitetura de VGG16 e VGG19



Fonte: Adaptado de Andrew e Santoso (2022).

A ilustração da arquitetura VGG16 e VGG19 apresenta uma sequência de camadas convolucionais e de *pooling*. Na VGG16, a estrutura típica começa com duas camadas convolucionais seguidas por uma camada de *pooling*. Esse padrão se repete, com o número de camadas convolucionais aumentando em blocos posteriores: dois blocos de duas camadas convolucionais cada, seguidos por três blocos de três camadas convolucionais cada. A VGG19 segue um padrão similar, mas com um bloco adicional de três camadas convolucionais, totalizando cinco blocos de convoluções antes das camadas totalmente conectadas.

A profundidade dessas arquiteturas permite que elas aprendam representações hierárquicas complexas das imagens, melhorando significativamente o desempenho em tarefas de classificação de imagens em grande escala. Os resultados obtidos por essas redes em competições, como o ILSVRC, foram uma taxa de erro de 7,3%, demonstrando uma precisão notável, estabelecendo novos padrões de referência para a comunidade de visão computacional (SIMONYAN; ZISSERMAN, 2015).

2.2.12 ResNet

As arquiteturas ResNet50, ResNet101 e ResNet152 são variantes da rede neural convolucional ResNet, introduzidas para resolver problemas de degradação em redes profundas. As ResNets utilizam conexões residuais, o que permite que gradientes fluam diretamente através da rede, mitigando o problema de desaparecimento de gradiente e facilitando a otimização de redes mais profundas (HE *et al*., 2016a).

A arquitetura básica da ResNet consiste em blocos residuais que incluem atalhos (*shortcuts*) que ignoram uma ou mais camadas. Este atalho adiciona a saída de uma camada anterior diretamente à saída de uma camada subsequente, formando a chamada "identidade de

atalho". Isso ajuda a preservar o gradiente e permite a construção de redes muito profundas sem a degradação de desempenho comum em arquiteturas tradicionais (HE *et al*., 2016a).

A ResNet50 consiste em 50 camadas, organizadas em um padrão específico de blocos residuais. Utiliza um design de bloco de gargalo (*bottleneck*), onde cada bloco residual consiste em três camadas convolucionais: 1x1, 3x3 e 1x1. As convoluções 1x1 reduzem e depois restauram as dimensões, enquanto a convolução 3x3 é usada como o gargalo que reduz a dimensionalidade de entrada e saída. A rede termina com uma camada de *pooling* global, seguida por uma camada totalmente conectada e uma camada *softmax* para a classificação (HE *et al*., 2016a).

A ResNet101 expande a arquitetura da ResNet50, utilizando um total de 101 camadas. Mantém o design do bloco de gargalo da ResNet50, mas aumenta o número de blocos residuais, permitindo uma maior profundidade e capacidade de aprendizado. Semelhante à ResNet50, termina com *pooling* global, uma camada totalmente conectada e uma camada *softmax* (HE *et al*., 2016a).

A ResNet152 é uma das versões mais profundas, consistindo em 152 camadas. Segue a mesma estrutura de blocos de gargalo das ResNets anteriores, mas com um número ainda maior de blocos residuais, oferecendo uma profundidade significativa que melhora o desempenho em tarefas complexas. Também termina com *pooling* global, uma camada totalmente conectada e uma camada *softmax* para classificação final (HE *et al*., 2016a).

A arquitetura da ResNet de forma geral é composta por uma camada convolucional inicial (7x7) com *stride* de 2, seguida por uma camada de *pooling* máximo (3x3) com *stride* de 2. No ResNet50 contém 16 blocos residuais organizados em quatro estágios (3, 4, 6, 3 blocos, respectivamente). No ResNet101 contém 33 blocos residuais organizados em quatro estágios (3, 4, 23, 3 blocos, respectivamente). E no ResNet152 contém 50 blocos residuais organizados em quatro estágios (3, 8, 36, 3 blocos, respectivamente). Nas camadas finais, se encontra um bloco de *pooling* global seguido por uma camada totalmente conectada e uma camada *softmax* (HE *et al*., 2016a).

As arquiteturas ResNet50, ResNet101 e ResNet152 são conhecidas por: mitigação de degradação, em que as conexões residuais permitem a construção de redes muito profundas sem a perda de desempenho, uma inovação crítica para a aprendizagem profunda; treinamento mais rápido, em que as conexões de atalho ajudam na convergência rápida durante o treinamento; e desempenho superior, em que nos *benchmarks* como o ImageNet, as ResNets superaram consistentemente as redes anteriores em termos de precisão de classificação (HE *et al*., 2016a).

O ResNet50V2 é uma versão aprimorada do ResNet50 original, desenvolvida para melhorar a propagação do gradiente e a eficiência do treinamento. Ele introduz a identidade como mapeamento no bloco residual e move a ativação e a normalização antes da adição do atalho, diferentemente do ResNet original, onde a ativação e a normalização são realizadas após a adição. Essas mudanças estruturais ajudam a manter um "caminho de informação limpo", facilitando a propagação do gradiente através de redes muito profundas (HE *et al*., 2016b).

A arquitetura do ResNet50V2 consiste em 50 camadas de profundidade, organizadas em blocos de três camadas (convolução 1x1, convolução 3x3 e convolução 1x1). A arquitetura específica é projetada para capturar diferentes níveis de características da imagem de entrada, mantendo uma eficiência computacional adequada para tarefas complexas de reconhecimento de imagem (HE *et al*., 2016b).

O ResNet101V2 segue o mesmo princípio arquitetônico do ResNet50V2, mas com um número maior de camadas, totalizando 101 camadas. A introdução de blocos residuais mais profundos permite que a rede aprenda representações mais complexas das características das imagens. Cada bloco residual no ResNet101V2 é projetado para preservar o fluxo de informações e gradientes, o que é essencial para a eficiência do treinamento em redes profundas. O uso de mapeamentos de identidade e pré-ativação em cada bloco residual é fundamental para manter a estabilidade do treinamento e melhorar a generalização (HE *et al*., 2016b).

O ResNet152V2 é ainda mais profundo, com 152 camadas, e segue os mesmos princípios de design do ResNet50V2 e ResNet101V2. Esta rede é projetada para tarefas extremamente complexas de reconhecimento de imagem, onde a profundidade adicional proporciona uma capacidade de modelagem substancialmente maior. A arquitetura mantém a estrutura de blocos residuais com mapeamentos de identidade e pré-ativação, garantindo que os gradientes possam se propagar eficientemente através das muitas camadas (HE *et al*., 2016b).

Na arquitetura dos modelos ResNet50V2, ResNet101V2 e ResNet152V2, cada bloco inclui camadas de convolução 1x1, 3x3 e 1x1, seguidas de normalização e ativação antes da adição do atalho de identidade, organizadas de maneira a manter um fluxo de informação consistente, facilitando a aprendizagem e a generalização. Esta estrutura modular permite um treinamento eficiente de redes muito profundas, mantendo a propagação dos gradientes intacta, o que é essencial para evitar problemas de desaparecimento do gradiente e para garantir que a rede possa ser treinada de forma eficaz (HE *et al*., 2016b).

As ResNetRS (ResNet-Rethinking-Scale) são uma família de redes neurais convolucionais que foram otimizadas para alcançar um equilíbrio entre precisão e eficiência

computacional (BELLO *et al*., 2021). A seguir, são detalhadas as diferentes variantes da ResNetRS.

A ResNetRS50 é uma versão otimizada da ResNet50 tradicional, incorporando técnicas modernas de treinamento e escalonamento. Esta arquitetura consiste em 50 camadas de redes neurais, otimizadas para serem 1.7x - 2.7x mais rápidas que as EfficientNets em TPUs e 2.1x - 3.3x mais rápidas em GPUs. Semelhante à ResNetRS50, a ResNetRS101 possui 101 camadas, proporcionando maior profundidade e capacidade de aprendizado, resultando em melhor desempenho em tarefas de reconhecimento de imagem mais complexas. Com 152 camadas, a ResNetRS152 é projetada para tarefas que exigem maior precisão e capacidade de captura de características mais profundas das imagens. A ResNetRS200, com 200 camadas, continua a tendência de aumentar a profundidade da rede para melhorar a precisão, especialmente útil em conjuntos de dados grandes e complexos. A ResNetRS270 vai além com 270 camadas, oferecendo uma rede ainda mais profunda e poderosa para tarefas de visão computacional. A ResNetRS350, com suas 350 camadas, é projetada para capturar ainda mais detalhes nas imagens, proporcionando precisão superior em tarefas avançadas de reconhecimento. A variante mais profunda, a ResNetRS420, com 420 camadas, é a mais avançada em termos de profundidade, oferecendo a maior capacidade de aprendizado e precisão entre todas as variantes da ResNetRS (BELLO *et al*., 2021).

A ResNetRS implementa várias melhorias em relação à arquitetura ResNet original, incluindo métodos modernos de treinamento e mudanças arquiteturais menores. Um exemplo é a incorporação das técnicas ResNet-D e *Squeeze-and-Excitation*, que aumentam a precisão da *top-1* no ImageNet de 79,0% para 83,4%. Além disso, estratégias aprimoradas de escalabilidade foram desenvolvidas, resultando em modelos que utilizam menos memória durante o treinamento e são significativamente mais rápidos tanto em TPUs quanto em GPUs (BELLO *et al*., 2021).

A escalabilidade das redes ResNetRS é baseada em duas estratégias principais: aumentar a profundidade do modelo quando há risco de *overfitting*, preferindo aumentar a largura em outros casos; e escalar a resolução da imagem mais lentamente do que em trabalhos anteriores. Essas estratégias foram derivadas de experimentos exaustivos que analisaram o desempenho dos modelos ao longo de várias escalas e durações de treinamento (BELLO *et al*., 2021).

Comparando as ResNetRS com as EfficientNets em uma curva de Pareto de velocidade-precisão, as ResNetRS demonstram ser 1,7x a 2,7x mais rápidas em TPUs e 2,1x a 3,3x mais rápidas em GPUs, mesmo tendo mais parâmetros e FLOPS (BELLO *et al*., 2021). Este

desempenho superior se deve, em parte, ao uso eficiente da memória e ao menor consumo de memória durante o treinamento, em comparação com as EfficientNets, que utilizam convoluções profundidade-separáveis com grandes ativações (BELLO *et al*., 2021).

2.2.13 InceptionV3

A arquitetura InceptionV3 é uma evolução da série de redes Inception, projetada para melhorar o desempenho em benchmarks de classificação, como o ILSVRC 2012. Esta arquitetura foi desenvolvida com base em várias técnicas e princípios apresentados em versões anteriores, como a InceptionV1 e InceptionV2, com aprimoramentos significativos para aumentar a eficiência e a precisão (SZEGEDY *et al.*, 2016).

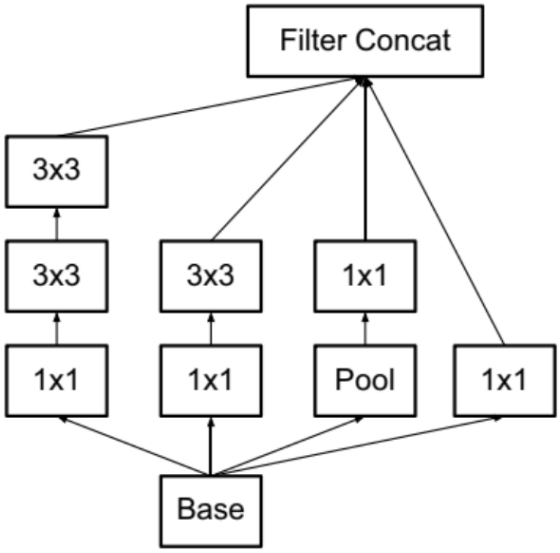
A estrutura da InceptionV3 segue uma série de camadas convolucionais e módulos Inception, cada um contribuindo para a extração de características em diferentes níveis de abstração (SZEGEDY *et al.*, 2016). A Tabela 1 resume a arquitetura da rede, com os tamanhos das saídas de cada módulo.

Tabela 1 - Arquitetura InceptionV3

Tipo	Tamanho do *Patch/Stride*	Tamanho da Entrada
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv (padded)	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	Mostrado na Figura 9	35×35×288
5×Inception	Mostrado na Figura 10	17×17×768
2×Inception	Mostrado na Figura 11	8×8×1280
pool	8×8	8×8×2048
linear	logits	1×1×2048
softmax	classifier	1×1×1000

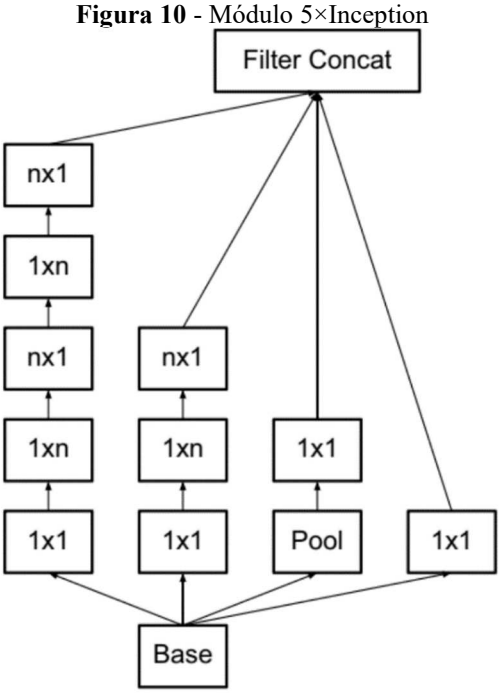
Fonte: Adaptado de Szegedy *et al.* (2016).

Figura 9 - Módulo 3×Inception



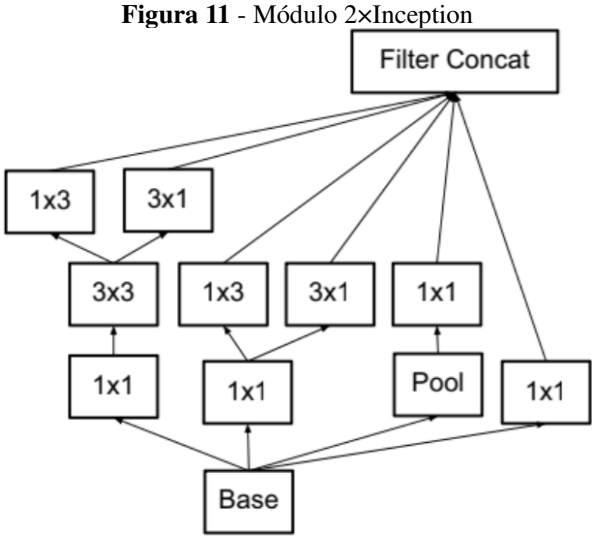
Fonte: Szegedy *et al.* (2016).

Figura 10 - Módulo 5×Inception



Fonte: Szegedy *et al.* (2016).

Figura 11 - Módulo 2×Inception



Fonte: Szegedy *et al.* (2016).

Nos primeiros níveis da rede, a InceptionV3 utiliza módulos Inception tradicionais. No nível de 35×35, há três módulos Inception com 288 filtros cada. Para reduzir o tamanho da grade e aumentar a eficiência computacional, a técnica de redução de *grid* é aplicada, reduzindo uma grade de 35×35 para 17×17 com 768 filtros. Após a redução da grade, cinco módulos Inception são aplicados com a fatoração das convoluções. Este processo continua reduzindo a grade para 8×8 com 1280 filtros. Nos níveis mais grossos (8×8), a InceptionV3 utiliza dois módulos Inception com bancos de filtros expandidos (SZEGEDY *et al.*, 2016).

As convoluções tradicionais de 7×7 são fatoradas em três convoluções de 3×3. Essa fatoração reduz o custo computacional enquanto mantém a eficácia da rede. A arquitetura inclui um classificador auxiliar, que é aplicado sobre a última camada de 17×17. A normalização em lote (*Batch Normalization*) é utilizada tanto nas camadas convolucionais quanto na camada totalmente conectada do classificador auxiliar, resultando em um ganho de precisão de 0,4% no Top-1 (SZEGEDY *et al.*, 2016).

A InceptionV3 é uma arquitetura poderosa e eficiente para tarefas de visão computacional, oferecendo melhorias significativas em termos de precisão e custo computacional em comparação com suas predecessoras. A aplicação de técnicas como a fatoração de convoluções e a redução de *grid* permite que a rede extraia características de maneira mais eficaz, mantendo uma estrutura gerenciável e eficiente para treinamento e inferência (SZEGEDY *et al.*, 2016).

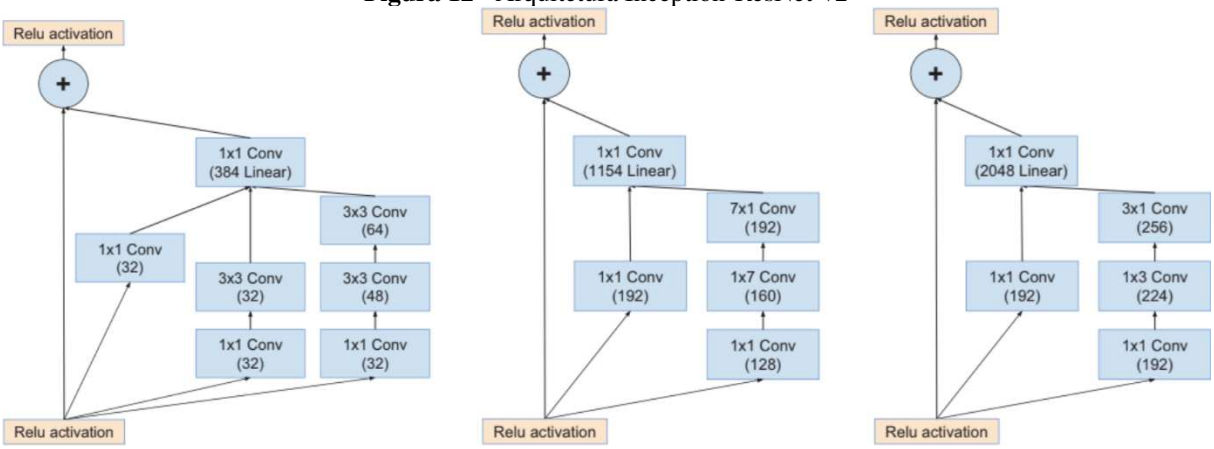
2.2.14 InceptionResNetV2

O Inception-ResNet-v2 é uma variação da arquitetura Inception que incorpora conexões residuais para melhorar o desempenho e a eficiência da rede. Esta combinação visa aproveitar as vantagens tanto da arquitetura Inception quanto das conexões residuais introduzidas por He *et al*. (2016a). A arquitetura Inception, inicialmente apresentada como GoogLeNet, passou por várias revisões para melhorar sua eficiência e desempenho. A introdução das conexões residuais nas redes profundas visou resolver o problema da degradação durante o treinamento de redes muito profundas, onde a adição de mais camadas poderia resultar em maior erro de treinamento e validação (SZEGEDY *et al*., 2017).

O Inception-ResNet-v2 combina blocos Inception com conexões residuais, substituindo a etapa de concatenação de filtros da arquitetura Inception original pelas conexões residuais. Isso permite que a rede mantenha sua eficiência computacional ao mesmo tempo em que aproveita os benefícios das conexões residuais, que facilitam o treinamento de redes profundas ao permitir que os gradientes fluam mais facilmente através da rede (SZEGEDY *et al*., 2017).

Para as versões residuais das redes Inception, foram utilizados blocos Inception mais econômicos. Cada bloco Inception é seguido por uma camada de expansão de filtros (convolução 1x1 sem ativação), que é usada para aumentar a dimensionalidade do banco de filtros antes da adição residual, a fim de corresponder à profundidade da entrada. Isso é necessário para compensar a redução de dimensionalidade induzida pelo bloco Inception (SZEGEDY *et al*., 2017). A Figura 12 ilustra os módulos de grade internos da rede Inception-ResNet-v2.

Figura 12 - Arquitetura Inception-ResNet-v2



Fonte: Szegedy *et al.* (2017).

Os módulos de grade 35×35, 17×17 e 8×8 são representados, respectivamente, pelos blocos Inception-A, Inception-B e Inception-C. Esses blocos são responsáveis por diferentes estágios de processamento dentro da rede, onde cada tipo de bloco é especializado em capturar diferentes tipos de características e padrões na imagem (SZEGEDY *et al*., 2017).

Os experimentos realizados pelos autores demonstraram que o Inception-ResNet-v2, juntamente com outras variantes como o Inception-v4, alcançou um desempenho de ponta no conjunto de validação do ImageNet, excedendo o estado da arte em desempenho de uma única imagem. No entanto, ao combinar múltiplos modelos (ensemble), os ganhos de desempenho não foram tão significativos quanto os observados nos modelos individuais, embora ainda tenham permitido alcançar um erro top-5 de 3,1% no conjunto de validação, estabelecendo um novo estado da arte (SZEGEDY *et al*., 2017).

2.2.15 DenseNet

As redes DenseNet são uma classe de arquiteturas de rede neural convolucional que introduzem uma forma de conectividade entre camadas. A característica principal das DenseNets é a conexão densa, onde cada camada recebe a entrada de todas as camadas anteriores e passa sua própria saída para todas as camadas subsequentes. Isso promove uma reutilização máxima das características extraídas e melhora a eficiência do fluxo de gradientes na rede, facilitando o treinamento de redes profundas (HUANG *et al*., 2017).

DenseNet121, DenseNet169 e DenseNet201 são variantes da arquitetura DenseNet, diferenciando-se principalmente pelo número de camadas densamente conectadas. Cada uma dessas variantes é projetada para diferentes níveis de complexidade e requisitos de memória, mas todas seguem o mesmo princípio de conectividade densa (HUANG *et al*., 2017).

Em uma DenseNet, a saída de cada camada é concatenada com as saídas de todas as camadas anteriores, em vez de serem somadas como nas ResNets. A equação que representa essa conexão densa é:

$$x_\ell = H_\ell\left(\left[x_0, x_1, \ldots, x_{\ell-1}\right]\right), \quad (1)$$

onde H_ℓ representa uma composição de operações (normalização de lote, ReLU e convolução) e $\left[x_0, x_1, \ldots, x_{\ell-1}\right]$ denota a concatenação das saídas das camadas anteriores (HUANG *et al*., 2017).

A arquitetura é dividida em blocos densos separados por camadas de transição. Cada bloco denso consiste em várias camadas de convolução que produzem características que são concatenadas. As camadas de transição, entre os blocos densos, realizam convoluções e operações de *pooling* para reduzir a dimensionalidade espacial das características. A taxa de crescimento *k* refere-se ao número de mapas de características que cada camada convolucional produz. Por exemplo, em DenseNet121, se *k*=32, então cada camada produz 32 novos mapas de características (HUANG *et al*., 2017).

A Tabela 2 resume a estrutura de DenseNet121, DenseNet169 e DenseNet201.

Tabela 2 - Estrutura de DenseNet121, DenseNet169 e DenseNet201

Camada	Tamanho de saída	DenseNet121 (k=32)	DenseNet169 (k=32)	DenseNet201 (k=32)
Convolução	112×112	7×7 conv, *stride* 2	7×7 conv, *stride* 2	7×7 conv, *stride* 2
Pooling	56×56	3×3 max pool, *stride* 2	3×3 max pool, *stride* 2	3×3 max pool, *stride* 2
Bloco Denso (1)	56×56	[1×1 conv, 3×3 conv]×6	[1×1 conv, 3×3 conv]×6	[1×1 conv, 3×3 conv]×6
Camada de Transição (1)	56×56	1×1 conv	1×1 conv	1×1 conv
	28×28	2×2 avg pool, *stride* 2	2×2 avg pool, *stride* 2	2×2 avg pool, *stride* 2
Bloco Denso (2)	28×28	[1×1 conv, 3×3 conv]×12	[1×1 conv, 3×3 conv]×12	[1×1 conv, 3×3 conv]×12
Camada de Transição (2)	28×28	1×1 conv	1×1 conv	1×1 conv
	14×14	2×2 avg pool, *stride* 2	2×2 avg pool, *stride* 2	2×2 avg pool, *stride* 2
Bloco Denso (3)	14×14	[1×1 conv, 3×3 conv]×24	[1×1 conv, 3×3 conv]×32	[1×1 conv, 3×3 conv]×48
Camada de Transição (3)	14×14	1×1 conv	1×1 conv	1×1 conv
	7×7	2×2 avg pool, *stride* 2	2×2 avg pool, *stride* 2	2×2 avg pool, *stride* 2
Bloco Denso (4)	7×7	[1×1 conv, 3×3 conv]×16	[1×1 conv, 3×3 conv]×32	[1×1 conv, 3×3 conv]×32
Camada de Classificação	1×1	7×7 global avg pool	7×7 global avg pool	7×7 global avg pool
		1000D fully connected, softmax	1000D fully connected, softmax	1000D fully connected, softmax

Fonte: Adaptado de Huang *et al*. (2017).

Esta estrutura densa permite que DenseNet121, DenseNet169 e DenseNet201 sejam mais eficientes em termos de parâmetros e melhorem a capacidade de reutilização de características, resultando em um desempenho superior em tarefas de reconhecimento de imagens (HUANG *et al*., 2017).

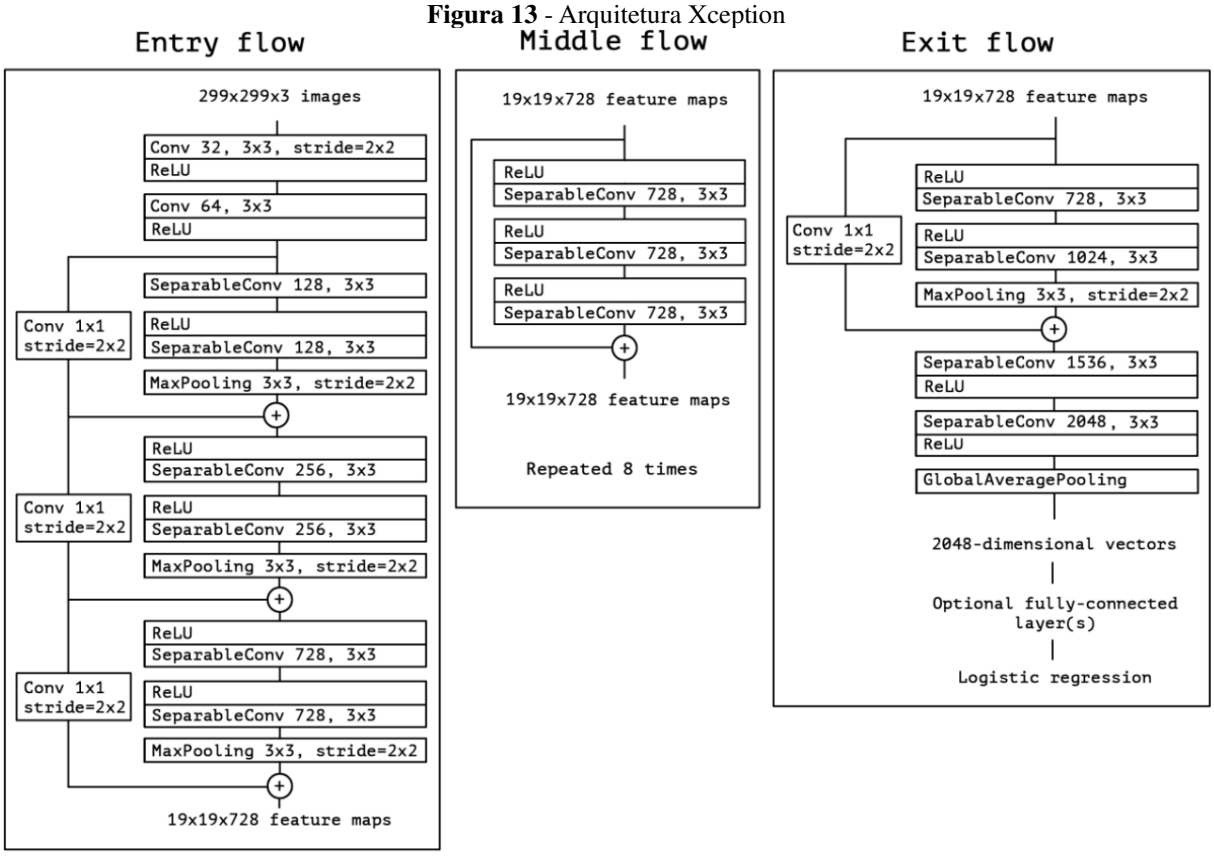
Essas arquiteturas são particularmente eficazes porque reduzem a redundância de características e permitem que a rede aprenda de forma mais eficiente, aproveitando a riqueza de informações das camadas anteriores. Ao contrário de outras arquiteturas que podem sofrer com o problema de gradientes desaparecendo, as DenseNets mantêm um fluxo robusto de gradientes durante o treinamento, facilitando a otimização de redes muito profundas (HUANG *et al*., 2017).

2.2.16 Xception

A arquitetura Xception é uma rede neural convolucional baseada inteiramente em camadas de convoluções separáveis por profundidade, que são mais eficientes em termos computacionais do que as convoluções tradicionais. Uma camada de convolução padrão opera aprendendo filtros em um espaço tridimensional: duas dimensões espaciais (largura e altura) e uma dimensão de canal. Isso significa que a convolução mapeia simultaneamente correlações espaciais e entre canais. No entanto, o Xception parte do pressuposto de que as correlações espaciais e as correlações entre canais podem ser aprendidas separadamente, levando ao uso de convoluções separáveis por profundidade.

A arquitetura Xception possui 36 camadas convolucionais que formam a base de extração de características da rede. As 36 camadas convolucionais são estruturadas em 14 módulos, todos com conexões residuais lineares ao seu redor, exceto os primeiros e últimos módulos. A arquitetura Xception é uma pilha linear de camadas de convoluções separáveis por profundidade com conexões residuais. Isso torna a arquitetura muito fácil de definir e modificar (CHOLLET, 2017). A Figura 13 ilustra a arquitetura Xception, onde os dados passam primeiro pelo fluxo de entrada, depois pelo fluxo intermediário, que é repetido oito vezes, e finalmente pelo fluxo de saída (CHOLLET, 2017).

Figura 13 - Arquitetura Xception



Fonte: Chollet (2017).

Todas as camadas de Convolução e Convolução Separável são seguidas por normalização em lote (*batch normalization*). Todas as camadas de Convolução Separável usam um multiplicador de profundidade de 1 (sem expansão de profundidade) (CHOLLET, 2017).

A arquitetura Xception consiste em três partes principais: fluxo de entrada (*entry flow*), fluxo intermediário (*middle flow*) e fluxo de saída (*exit flow*). Os dados passam inicialmente pelo fluxo de entrada, que processa as características básicas. Em seguida, passam pelo fluxo intermediário, onde o processo é repetido oito vezes para refinar as características. Finalmente, os dados passam pelo fluxo de saída, onde as características finais são extraídas para a classificação (CHOLLET, 2017).

Cada camada de Convolução e Convolução Separável é seguida por uma normalização em lote para manter a estabilidade do treinamento. A Convolução Separável por Profundidade permite que a rede seja mais eficiente, processando cada canal de entrada separadamente, seguido por uma combinação linear dos resultados (CHOLLET, 2017).

Essa estrutura torna a arquitetura Xception mais eficiente em termos de parâmetros e computação, ao mesmo tempo que melhora a precisão na classificação de imagens comparada à Inception V3 (CHOLLET, 2017).

2.2.17 MobileNet

MobileNet é uma classe de modelos eficientes voltada para aplicações de visão em dispositivos móveis e embarcados. MobileNets são baseadas em uma arquitetura simplificada que utiliza convoluções separáveis em profundidade para construir redes neurais profundas e leves. Convoluções separáveis em profundidade são um tipo de convolução que divide uma convolução padrão em duas camadas: uma convolução de profundidade (*depthwise convolution*) e uma convolução 1x1 (*pointwise convolution*). Foram introduzidos dois hiperparâmetros globais simples que permitem equilibrar de forma eficiente entre latência e precisão. Esses hiperparâmetros permitem ao desenvolvedor escolher o tamanho adequado do modelo para sua aplicação, com base nas restrições do problema (HOWARD *et al*., 2017).

A convolução de profundidade aplica um único filtro a cada canal de entrada, enquanto a convolução *pointwise* combina as saídas dessas convoluções em profundidade. Essa fatoração reduz drasticamente a computação e o tamanho do modelo (HOWARD *et al*., 2017).

Uma camada convolucional padrão aplica filtros e combina entradas em um novo conjunto de saídas em um único passo. A convolução separável em profundidade divide esse processo em duas camadas: uma camada para filtrar e outra para combinar, o que reduz significativamente os custos computacionais (HOWARD *et al*., 2017).

Essas convoluções, quando combinadas, são chamadas de convoluções separáveis em profundidade, que são altamente eficientes em termos computacionais. A MobileNet é construída com base em convoluções separáveis em profundidade, exceto pela primeira camada, que é uma convolução completa. A arquitetura é definida de forma a explorar topologias de rede para encontrar a melhor configuração. Todas as camadas são seguidas por normalização em batch e não linearidade ReLU, exceto a última camada totalmente conectada, que alimenta uma camada softmax para classificação. A MobileNet utiliza menos regularização e técnicas de aumento de dados, já que modelos pequenos têm menos problemas de *overfitting* (HOWARD *et al*., 2017).

A MobileNetV2 também é uma arquitetura de rede neural desenvolvida especificamente para dispositivos móveis e ambientes com recursos limitados. Ela avança o estado da arte para modelos de visão computacional móveis, reduzindo significativamente o número de operações e a memória necessária, mantendo a mesma precisão. A principal contribuição da MobileNetV2 é um módulo de camada inovador: o residual invertido com gargalo linear. Este módulo começa com uma representação comprimida de baixa dimensão, expande-a para alta dimensão usando

convoluções leves em profundidade e, em seguida, projeta novamente para uma representação de baixa dimensão com uma convolução linear (SANDLER *et al*., 2018).

A estrutura da MobileNetV2 baseia-se em conexões de atalho entre camadas de gargalo finas. A camada de expansão intermediária utiliza convoluções em profundidade para filtrar características, introduzindo não linearidades. Além disso, é importante remover as não linearidades nas camadas estreitas para manter o poder de representação. Isso melhora o desempenho, permitindo desacoplar os domínios de entrada/saída da expressividade da transformação, facilitando a análise adicional da arquitetura (SANDLER *et al*., 2018).

As camadas de gargalo residual invertido permitem uma implementação particularmente eficiente em termos de memória, o que é crucial para aplicativos móveis. Durante a inferência, a memória é utilizada de forma otimizada, reduzindo a necessidade de acesso à memória principal em muitos designs de hardware embarcado, que possuem pequenas quantidades de memória cache controlada por software (SANDLER *et al*., 2018).

A Tabela 3 descreve a sequência de camadas da MobileNetV2, onde cada linha representa uma sequência de uma ou mais camadas idênticas (módulo *stride*), repetidas n vezes. Todas as camadas na mesma sequência têm o mesmo número de canais de saída c. A primeira camada de cada sequência tem um *stride* s e todas as outras usam *stride* 1. Todas as convoluções espaciais usam *kernels* 3 × 3. O fator de expansão t é sempre aplicado ao tamanho da entrada (SANDLER *et al*., 2018).

Tabela 3 - Sequência de camadas da MobileNetV2

Tamanho de entrada	Operador	t	c	n	s
224^2 x 3	conv2d-	32	1	2	-
112^2 x 32	bottleneck	1	16	1	1
112^2 x 16	bottleneck	6	24	2	2
56^2 x 24	bottleneck	6	32	3	2
28^2 x 32	bottleneck	6	64	4	2
14^2 x 64	bottleneck	6	96	3	1
14^2 x 96	bottleneck	6	160	3	2
7^2 x 160	bottleneck	6	320	1	1
7^2 x 320	conv2d 1x1	-	1280	1	1
7^2 x 1280	avgpool 7x7	-	-	1	-
1^2 x 1280	conv2d 1x1	-	k	-	-

Fonte: Sandler *et al.* (2018).

A MobileNetV2 é comparada com outras redes como MobileNetV1, ShuffleNet e NasNet-A em termos de precisão e eficiência computacional. A Tabela 4 mostra as estatísticas de desempenho de alguns modelos selecionados.

Tabela 4 - Estatística de desempenho de MobileNetV1, ShuffleNet(1.5), ShuffleNet(x2), NasNet-A, MobileNetV2 e MobileNetV2(1.4)

Rede	Top 1	Parâmetros	MAdds	CPU (ms)
MobileNetV1	70,6	4,2M	575M	113
ShuffleNet (1.5)	71,5	**3,4M**	292M	-
ShuffleNet (x2)	73,7	5,4M	524M	-
NasNet-A	74,0	5,3M	564M	183
MobileNetV2	**72,0**	**3,4M**	**300M**	**75**
MobileNetV2 (1.4)	**74,7**	6,9M	585M	**143**

Fonte: Sandler *et al.* (2018).

A MobileNetV2 não apenas mantém uma alta precisão, mas também é altamente eficiente em termos de número de operações (MAdds) e tempo de execução, tornando-se ideal para dispositivos móveis e aplicações em tempo real (SANDLER *et al*., 2018).

O bloco de convolução da MobileNetV2 usa convoluções em profundidade seguidas por convoluções ponto a ponto (1x1) com conexões lineares, otimizando tanto a eficiência quanto a representatividade das características extraídas (SANDLER *et al*., 2018).

A MobileNetV3 é uma evolução das versões MobileNetV1 e MobileNetV2 otimizada usando uma combinação de busca de arquitetura NAS consciente de hardware e outras técnicas algorítmicas. A MobileNetV3 introduz dois modelos: MobileNetV3Large e MobileNetV3Small, cada um voltado para diferentes casos de uso – alto recurso e baixo recurso, respectivamente. Ambos os modelos são particularmente adequados para tarefas como classificação, detecção e segmentação (HOWARD *et al*., 2019).

Na MobileNetV3, foram feitos vários avanços importantes e utiliza-se de várias técnicas. A NAS consciente de plataforma otimiza a arquitetura para dispositivos móveis ao encontrar estruturas de rede eficientes. Especificamente, a técnica NAS consciente de plataforma garante que o modelo atinja um equilíbrio entre latência e precisão, considerando as limitações de hardware das CPUs móveis. O algoritmo NetAdapt ajusta a rede camada por camada para reduzir a latência sem comprometer a precisão. O processo itera por propostas de arquitetura e seleciona as que oferecem os melhores *trade-offs* entre latência e precisão. Os módulos *Squeeze-and-Excite* permitem que o modelo se concentre dinamicamente nos recursos

relevantes. Na MobileNetV3, o tamanho do gargalo do *Squeeze-and-Excite* é ajustado para aumentar a eficiência (HOWARD *et al*., 2019).

A MobileNetV3 introduz uma nova função de ativação chamada *hard-swish* (*h-swish*), uma aproximação computacionalmente eficiente da função *swish*. Esta função utiliza operações baseadas em ReLU para reduzir a carga computacional, especialmente importante em ambientes móveis onde as capacidades de hardware são limitadas (HOWARD *et al*., 2019).

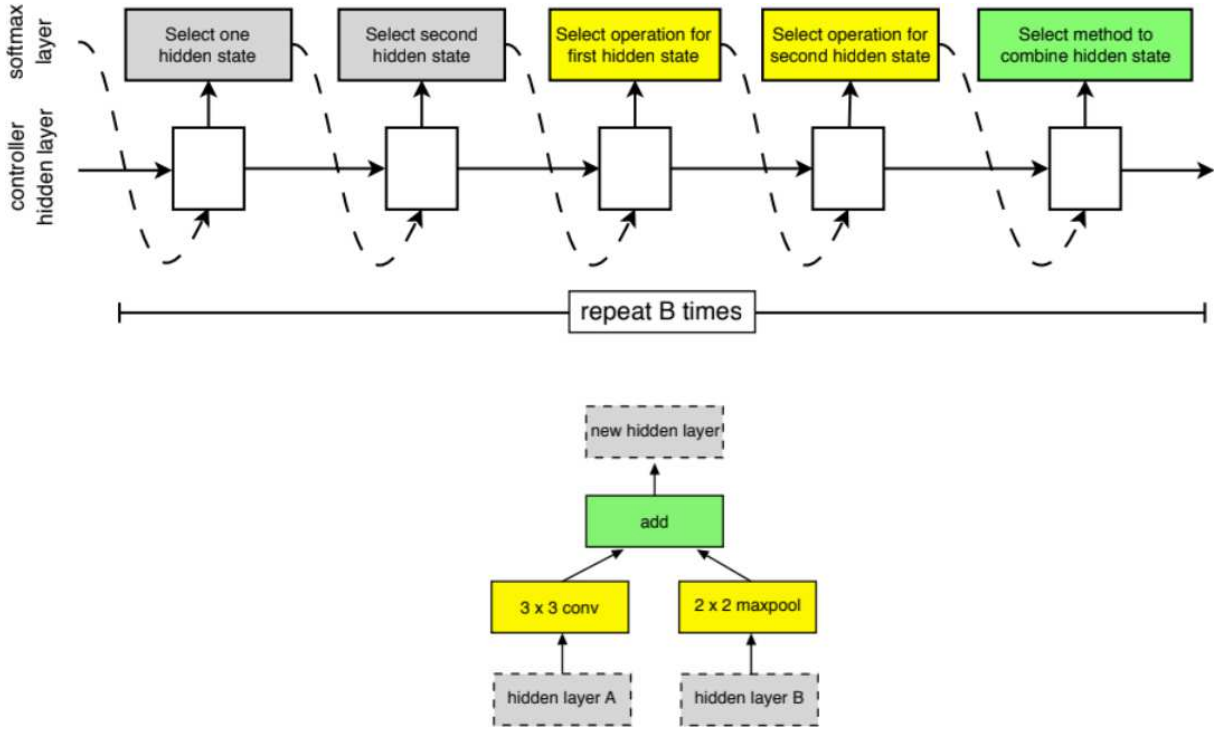
2.2.18 NASNet

NASNet (*Neural Architecture Search Network*) é uma arquitetura de rede neural convolucional que é composta de duas principais estruturas de células convolucionais: a célula normal, que mantém as dimensões de entrada; e a célula de redução, que reduz as dimensões da entrada pela metade. Cada célula é composta por blocos que contêm uma série de operações convolucionais e de pooling (ZOPH *et al*., 2018).

A busca pela arquitetura ótima é feita utilizando uma RNN (Rede Neural Recorrente). A RNN gera diferentes arquiteturas que são avaliadas e refinadas durante o treinamento. O processo inclui a predição recursiva, em que o controlador prevê uma série de operações e combinações de estados ocultos, e reforço, em que a performance de cada arquitetura gerada é avaliada e utilizada para ajustar o controlador (ZOPH *et al*., 2018).

A Figura 14 mostra o processo de construção de uma célula convolucional, em que se realiza a seleção de dois estados ocultos iniciais, aplica operações nesses estados (como convoluções 3x3, *pooling*, etc.), combina os resultados das operações e depois repete o processo por um número definido de blocos.

Figura 14 - Arquitetura do modelo de controlador para construção recursiva de um bloco de uma célula convolucional



Fonte: Zoph *et al*. (2018).

Cada célula recebe como entrada dois estados ocultos iniciais h_i e h_i-1 que são as saídas de duas células nas duas camadas inferiores anteriores ou a imagem de entrada. O controlador RNN prevê recursivamente o resto da estrutura da célula convolucional, dados esses dois estados ocultos iniciais. As predições do controlador para cada célula são agrupadas em blocos B, onde cada bloco possui 5 etapas de predição feitas por 5 classificadores *softmax* distintos correspondentes a escolhas discretas dos elementos de um bloco:

Etapa 1. Selecione um estado oculto de h_i, h_i-1 ou o conjunto de estados ocultos criados em blocos anteriores.

Etapa 2. Selecione um segundo estado oculto nas mesmas opções da Etapa 1.

Etapa 3. Selecione uma operação para aplicar ao estado oculto selecionado na Etapa 1.

Etapa 4. Selecione uma operação para aplicar ao estado oculto selecionado na Etapa 2.

Etapa 5. Selecione um método para combinar as saídas das Etapas 3 e 4 para criar um novo estado oculto.

O algoritmo anexa o estado oculto recém-criado ao conjunto de estados ocultos existentes como uma entrada potencial em blocos subsequentes. O controlador RNN repete as 5 etapas de predição acima B vezes correspondentes aos blocos B em uma célula convolucional (ZOPH *et al*., 2018).

Dentro de NasNet, existem duas variantes: NASNetMobile e NASNetLarge. Ambas as variantes utilizam a mesma abordagem de arquitetura com diferentes configurações para atender a diferentes requisitos de recursos e precisão. NASNetMobile é um modelo otimizado para dispositivos móveis, mantendo um bom balanço entre precisão e eficiência, e o NASNetLarge foi projetado para oferecer alta precisão em tarefas de classificação de imagem com maior custo computacional (ZOPH *et al*., 2018).

2.2.19 EfficientNet

A família EfficientNet começou com o modelo base EfficientNetB0. Desenvolvido utilizando busca neural multiobjetiva, ele otimiza tanto a precisão quanto o número de operações de ponto flutuante (FLOPS). O bloco principal de construção do EfficientNetB0 é o MBConv invertido, ao qual foi adicionada a otimização de squeeze-and-excitation para melhorar a eficiência e a precisão do modelo (TAN; LE, 2019).

A arquitetura do EfficientNetB0 consiste em várias camadas organizadas em estágios. Cada estágio é definido por um operador específico, resolução de entrada e saída, número de canais e número de camadas. O EfficientNetB0 foi projetado para ser um modelo de tamanho móvel, o que o torna eficiente em termos de FLOPS e adequado para dispositivos com recursos limitados (TAN; LE, 2019).

Os modelos EfficientNetB1 a B7 são escalados a partir do EfficientNetB0 usando um método de escalonamento composto. Este método utiliza um coeficiente composto (φ) para aumentar uniformemente a profundidade, largura e resolução da rede. Os valores de escalonamento (α, β, γ) foram determinados através de uma pequena busca em grade para o EfficientNetB0 e aplicados consistentemente aos modelos subsequentes (TAN; LE, 2019).

O método de escalonamento composto funciona da seguinte forma:

- Profundidade (d): α^{φ}
- Largura (w): β^{φ}
- Resolução (r): γ^{φ}

onde α, β, γ são constantes e φ é o coeficiente que controla a quantidade de recursos adicionais disponíveis para o escalonamento do modelo. Para os modelos EfficientNet, $\alpha = 1{,}2$, $\beta = 1{,}1$, e $\gamma = 1{,}15$, sob a restrição de que $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$. Este método garante que o aumento de recursos resulte em um aumento balanceado de profundidade, largura e resolução, melhorando a precisão e a eficiência do modelo. Cada variante do EfficientNet apresenta melhorias significativas em termos de precisão e eficiência computacional (TAN; LE, 2019).

EfficientNetB1 tem maior precisão que o ResNet-152, com 7,8 milhões de parâmetros e 0,7 bilhões de FLOPS. EfficientNetB2 supera modelos como o Inception-v4, utilizando apenas 9,2 milhões de parâmetros e 1,0 bilhão de FLOPS. EfficientNetB3 apresenta precisão superior ao ResNeXt-101 com uma fração dos FLOPS. EfficientNetB4, com 19 milhões de parâmetros, compete com modelos complexos como SENet e NASNet-A. EfficientNetB5, com 30 milhões de parâmetros, oferece precisão comparável ao AmoebaNet-C. EfficientNetB6 oferece alta precisão com 43 milhões de parâmetros e 19 bilhões de FLOPS. E EfficientNetB7 alcança a maior precisão, superando modelos como GPipe, com 66 milhões de parâmetros e 37 bilhões de FLOPS (TAN; LE, 2019).

A arquitetura de EfficientNetB0 a B7 é ilustrada como uma série de camadas convolucionais organizadas em estágios, onde cada estágio aumenta progressivamente em profundidade, largura e resolução. Isso permite que os modelos capturem mais detalhes e informações relevantes das imagens de entrada, melhorando a precisão das previsões. As ilustrações das arquiteturas mostram como cada versão escalada do EfficientNet aumenta esses parâmetros de forma balanceada para otimizar a performance (TAN; LE, 2019).

Os modelos EfficientNetV2 foram desenvolvidos para oferecer um treinamento mais rápido e maior eficiência de parâmetros, mantendo ou melhorando a precisão dos modelos anteriores da série EfficientNet. A seguir, apresenta-se uma descrição detalhada dos modelos EfficientNetV2B0, EfficientNetV2B1, EfficientNetV2B2, EfficientNetV2B3, EfficientNetV2L, EfficientNetV2M e EfficientNetV2S, explicando suas características e melhorias (TAN; LE, 2021).

EfficientNetV2B0, EfficientNetV2B1, EfficientNetV2B2 e EfficientNetV2B3 são variações incrementais que aumentam progressivamente em termos de complexidade e capacidade. EfficientNetV2B0 serve como modelo base. Utiliza tanto MBConv quanto Fused-MBConv, oferecendo um equilíbrio entre eficiência computacional e precisão. As operações Fused-MBConv combinam convoluções de profundidade com convoluções padrão para melhorar a eficiência nos estágios iniciais da rede.

EfficientNetV2B1 e EfficientNetV2B2 aumentam a profundidade da rede e o tamanho dos filtros, resultando em uma melhoria na precisão ao custo de um aumento moderado nos parâmetros e nas FLOPS. EfficientNetV2B3 aumenta ainda mais a profundidade e a complexidade, atingindo uma precisão superior comparada aos modelos anteriores, com um aumento proporcional nos parâmetros e FLOPS. EfficientNetV2L, EfficientNetV2M e EfficientNetV2S são projetados para aplicações que exigem diferentes *trade-offs* entre precisão, tempo de treinamento e tamanho do modelo. EfficientNetV2S é um modelo pequeno, mas

eficiente, ideal para dispositivos com recursos limitados. Ele mantém uma alta precisão enquanto reduz significativamente os parâmetros e FLOPS em comparação com os modelos maiores. EfficientNetV2M representa um equilíbrio entre tamanho e precisão, sendo adequado para uma variedade de aplicações que necessitam de um bom desempenho sem exigir hardware muito potente . EfficientNetV2L é o maior e mais preciso dos modelos V2, ideal para aplicações que podem aproveitar o aumento de precisão oferecido por um modelo com mais parâmetros e maior complexidade computacional (TAN; LE, 2021).

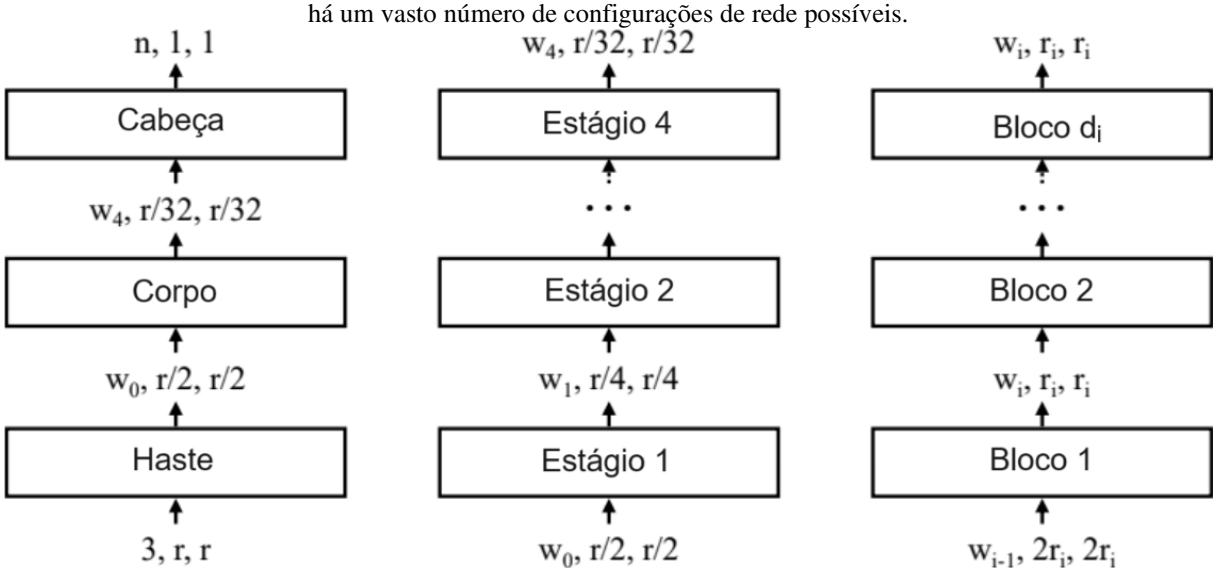
A arquitetura dos modelos EfficientNetV2 faz uso extensivo dos blocos MBConv e Fused-MBConv. MBConv é um bloco de convolução que combina convoluções de profundidade e convoluções de ponto para aumentar a eficiência da rede, e Fused-MBConv combina operações de convolução padrão com convoluções de profundidade em um único bloco, melhorando a eficiência computacional e a velocidade de treinamento nos estágios iniciais da rede (TAN; LE, 2021).

A arquitetura segue um design escalável que permite ajustar a profundidade (número de camadas), largura (número de filtros por camada) e resolução da entrada de imagem de maneira sistemática. Esta abordagem de escalonamento composto é uma das principais inovações que permitem aos modelos EfficientNetV2 alcançar uma alta eficiência de parâmetros e desempenho superior (TAN; LE, 2021).

2.2.20 RegNet

As redes RegNetX e RegNetY foram desenvolvidas como parte de um estudo sobre design de espaços de rede, visando otimizar o desempenho em diversas métricas de complexidade e eficiência de treinamento. Estas redes são parametrizadas por várias características, como a largura do grupo (g), profundidade (d), largura inicial (w_0), incremento de largura (w_a), e parâmetros de quantização (wm). As redes RegNetX e RegNetY diferem principalmente na inclusão de blocos *Squeeze-and-Excitation* nas RegNetY, o que aumenta ligeiramente a complexidade, mas melhora o desempenho (RADOSAVOVIC *et al.*, 2020). A Figura 15 mostra a estrutura geral das redes RegNet.

Figura 15 - Estrutura geral de rede para modelos RegNet. (a) Cada rede consiste em três partes principais: uma haste inicial (convolução 3×3 com *stride* dois e 32 canais de saída), seguida pelo corpo da rede que executa a maior parte da computação e, em seguida, uma cabeça (agrupamento médio seguido por uma camada totalmente conectada) que prevê n classes de saída. (b) O corpo da rede é composto por uma sequência de estágios que operam com resolução r_i progressivamente reduzida. (c) Cada estágio é composto por uma sequência de blocos idênticos, exceto o primeiro bloco que utiliza convolução com *stride* dois. Embora a estrutura geral seja simples, há um vasto número de configurações de rede possíveis.



(a) Rede (b) Corpo (c) Estágio i

Fonte: Adaptado de Radosavovic *et al*. (2020).

A estrutura da rede inclui elementos como o número de blocos (profundidade da rede), larguras de bloco (número de canais) e outros parâmetros de bloco, como taxas de gargalo ou larguras de grupo. O corpo da rede consiste em 4 estágios operando com resolução progressivamente reduzida. Cada estágio consiste em uma sequência de blocos idênticos. No total, para cada estágio *i*, os graus de liberdade incluem o número de blocos *di*, a largura do bloco w_i e quaisquer outros parâmetros do bloco.

A Tabela 5 mostra a comparação entre os modelos RegNetX, e a Tabela 6 mostra entre os modelos RegNetY.

Tabela 5 - Comparação de Modelos RegNetX

Modelos	FLOPS	Parâmetros	Ativações	Tamanho do batch	Tempo de inferência	Tempo de treino	Erro top-1
RegNetX002	0,2B	2,7M	2,2M	1024	10ms	2,8h	31,1%
RegNetX004	0,4B	5,2M	3,1M	1024	15ms	3,9h	27,3%
RegNetX006	0,6B	6,2M	4,0M	1024	17ms	4,4h	25,9%
RegNetX008	0,8B	7,3M	5,1M	1024	21ms	5,7h	24,8%
RegNetX016	1,6B	9,2M	7,9M	1024	33ms	8,7h	23,0%
RegNetX032	3,2B	15,3M	11,4M	512	57ms	14,3h	21,7%
RegNetX040	4,0B	22,1M	12,2M	512	69ms	17,1h	21,4%
RegNetX064	6,4B	26,2M	16,4M	512	92ms	23,5h	20,8%
RegNetX080	8,0B	39,6M	14,1M	512	94ms	22,6h	20,7%
RegNetX120	12,1B	46,1M	21,4M	512	137ms	32,9h	20,3%
RegNetX160	15,9B	54,3M	25,5M	512	168ms	39,7h	20,0%
RegNetX320	31,7B	107,8M	36,3M	256	318ms	76,9h	19,5%

Fonte: Adaptado de Radosavovic *et al*. (2020).

Tabela 6 - Comparação de Modelos RegNetY

Modelos	FLOPS	Parâmetros	Ativações	Tamanho do batch	Tempo de inferência	Tempo de treino	Erro top-1
RegNetY002	0,2B	3,2M	2,2M	1024	11ms	3,1h	29,6%
RegNetY004	0,4B	4,3M	3,9M	1024	19ms	5,1h	25,9%
RegNetY006	0,6B	6,1M	4,3M	1024	19ms	5,2h	24,5%
RegNetY008	0,8B	6,3M	5,2M	1024	22ms	6,0h	23,7%
RegNetY016	1,6B	11,2M	8,0M	1024	39ms	10,1h	22,0%
RegNetY032	3,2B	19,4M	11,3M	512	67ms	16,5h	21,0%
RegNetY040	4,0B	20,6M	12,3M	512	68ms	16,8h	20,6%
RegNetY064	6,4B	30,6M	16,4M	512	104ms	26,1h	20,1%
RegNetY080	8,0B	39,2M	18,0M	512	113ms	28,1h	20,1%
RegNetY120	12,1B	50,4M	24,1M	512	162ms	40,7h	19,8%
RegNetY160	16,0B	83,6M	28,1M	512	210ms	52,4h	19,7%
RegNetY320	31,7B	145,1M	44,3M	256	400ms	99,6h	19,5%

Fonte: Adaptado de Radosavovic *et al*. (2020).

Ambas as famílias de redes, RegNetX e RegNetY, demonstram que é possível criar arquiteturas de rede eficientes e escaláveis utilizando princípios de projeto baseados em quantização e ajuste fino de parâmetros. As redes RegNetY, com a inclusão dos blocos SE, tendem a apresentar um desempenho ligeiramente melhor, mas com um aumento correspondente na complexidade (RADOSAVOVIC *et al.*, 2020).

2.2.21 ConvNeXt

O modelo ConvNeXt foi desenvolvido para melhorar a performance dos ConvNets tradicionais, mantendo a simplicidade estrutural. As variantes ConvNeXtTiny, ConvNeXtSmall, ConvNeXtBase, ConvNeXtLarge e ConvNeXtXLarge apresentam diferentes tamanhos e capacidades, ajustando-se a diversas necessidades computacionais e de desempenho (LIU *et al.*, 2022).

O ConvNeXtTiny é projetado para ser leve, com um número reduzido de parâmetros e menor complexidade computacional, mantendo uma performance robusta em tarefas de classificação de imagens. Ele é adequado para aplicações com restrições de recursos, como dispositivos móveis (LIU *et al.*, 2022).

Em comparação com o ConvNeXtTiny, o ConvNeXtSmall tem um aumento moderado no número de parâmetros e na complexidade computacional, e oferece um equilíbrio entre eficiência e precisão, tornando-o adequado para uma ampla gama de aplicações (LIU *et al.*, 2022).

O ConvNeXtBase representa um ponto intermediário, com capacidade e complexidade maiores que as versões Tiny e Small, permitindo maior precisão em tarefas de visão computacional sem sacrificar a eficiência (LIU *et al.*, 2022).

O ConvNeXtLarge foi projetado para aplicações que demandam alta precisão, possui um número significativamente maior de parâmetros, oferecendo uma performance superior em tarefas complexas de classificação e segmentação de imagens (LIU *et al.*, 2022).

O ConvNeXtXLarge é o mais robusto da família ConvNeXt, com o maior número de parâmetros e a maior complexidade computacional. É ideal para aplicações que requerem máxima precisão e têm acesso a recursos computacionais abundantes (LIU *et al.*, 2022).

A arquitetura dos modelos ConvNeXt segue uma estrutura hierárquica, inspirada nos avanços recentes dos Transformers, mas mantendo a essência dos ConvNets. Cada modelo da família ConvNeXt é construído a partir de blocos padrão de convolução, ajustados para otimizar a eficiência e a precisão (LIU *et al.*, 2022).

O ConvNeXtTiny utiliza blocos convolucionais básicos com configurações ajustadas para minimizar a complexidade computacional. O ConvNeXtSmall expande a estrutura do ConvNeXtTiny com mais filtros e camadas convolucionais, melhorando a capacidade de extração de características. O ConvNeXtBase adiciona mais profundidade e largura à rede, permitindo uma melhor captura de padrões complexos nas imagens. O ConvNeXtLarge aumenta ainda mais o número de parâmetros e a profundidade da rede, otimizando a precisão

em tarefas de visão computacional. O ConvNeXtXLarge maximiza a capacidade da rede com um número substancial de parâmetros, oferecendo a melhor performance dentro da família ConvNeXt.

As configurações específicas de canais (C) e blocos (B) para os modelos ConvNeXt são as seguintes:

ConvNeXtTiny: C = (64, 128, 256, 512), B = (3, 3, 9, 3)

ConvNeXtSmall: C = (96, 192, 384, 768), B = (3, 3, 9, 3)

ConvNeXtBase: C = (128, 256, 512, 1024), B = (3, 3, 27, 3)

ConvNeXtLarge: C = (192, 384, 768, 1536), B = (3, 3, 27, 3)

ConvNeXtXLarge: C = (256, 512, 1024, 2048), B = (3, 3, 27, 3)

Essas configurações demonstram a escalabilidade e a modularidade da arquitetura ConvNeXt, permitindo ajustes precisos para diferentes necessidades de desempenho e eficiência (LIU *et al.*, 2022).

A seção seguinte abordará a redução de dimensionalidade, que é um processo realizado para obter os atributos mais relevantes para descrever um objeto dentro de um conjunto de dados.

2.3 Redução de dimensionalidade

Um problema de estimação da qualidade de atributos é uma questão importante no aprendizado de máquina. Em muitos problemas de aprendizado, existem várias características potenciais para descrever cada objeto de entrada. Existe uma tarefa que realiza uma seleção ou uma extração de características mais relevantes para descrever um objeto, a fim de reduzir a dimensionalidade. Para tomar uma decisão sobre quais características escolher e quais descartar, é necessário ter um método confiável e praticamente eficiente para estimar sua relevância (ROBNIK-SIKONJA; KONONENKO, 2003).

O processo de redução de dimensionalidade é importante para mitigar a ocorrência do fenômeno conhecido como "maldição da dimensionalidade". A maldição da dimensionalidade refere-se a diversos fenômenos que surgem ao analisar dados em espaços de alta dimensão, onde o número de dimensões dos dados pode impactar negativamente o desempenho dos algoritmos de aprendizado de máquina. Este conceito foi inicialmente abordado por Richard Bellman na década de 1960 e destaca os desafios que surgem ao lidar com grandes volumes de dados em múltiplas dimensões (DEBIE; SHAFI, 2017).

Em espaços de alta dimensão, os dados tornam-se esparsos, o que significa que a densidade dos pontos de dados diminui drasticamente à medida que o número de dimensões aumenta. Isso afeta a capacidade dos algoritmos de aprendizado em encontrar padrões ou relações significativas nos dados. Além disso, o volume de um hipercubo (uma extensão multidimensional de um cubo) aumenta exponencialmente com o número de dimensões, fazendo com que a maior parte do volume esteja próxima da superfície. Consequentemente, os dados tendem a se concentrar nas bordas do espaço dimensional, tornando difícil a identificação de estruturas ou clusters significativos (DEBIE; SHAFI, 2017).

Outro problema associado à maldição da dimensionalidade é que, em espaços de alta dimensão, a distância entre os pontos de dados se torna cada vez mais uniforme. Em outras palavras, a diferença relativa entre a distância do ponto mais próximo e a do ponto mais distante diminui. Isso pode degradar o desempenho de algoritmos que dependem de medidas de distância, como KNN e SVM, pois a capacidade de discriminar entre pontos próximos e distantes é reduzida (DEBIE; SHAFI, 2017).

Para mitigar os efeitos da maldição da dimensionalidade, várias abordagens são sugeridas. Métodos de redução de dimensionalidade, como PCA (PEARSON, 1901) e LDA (FISHER, 1936), são frequentemente usados para transformar dados de alta dimensão em um espaço de menor dimensão que preserva a maior parte da variabilidade ou da discriminação dos dados originais. Além disso, técnicas de regularização podem ser aplicadas para prevenir o *overfitting* em espaços de alta dimensão (DEBIE; SHAFI, 2017).

Portanto, a maldição da dimensionalidade é um desafio significativo no campo do aprendizado de máquina e da mineração de dados. Compreender e abordar esses desafios é crucial para o desenvolvimento de algoritmos eficazes que possam lidar com dados de alta dimensão de maneira eficiente e precisa (DEBIE; SHAFI, 2017).

A Análise de Componentes Principais, ou em inglês *Principle Components Analysis* (PCA), criada por Pearson (1901), é uma técnica estatística utilizada para reduzir a dimensionalidade de um conjunto de dados, ao mesmo tempo que retém a maior quantidade possível da variação presente nos dados originais fazendo uma projeção linear desses dados, minimizando o erro de reconstrução. A PCA é amplamente utilizada em áreas como reconhecimento de padrões, compressão de dados e visualização de dados multidimensionais (JOLLIFFE, 2002).

O procedimento básico da PCA envolve os seguintes passos: normalização dos dados, em que os dados são normalizados para que cada variável tenha média zero e variância um, garantindo que todas as variáveis contribuam igualmente para a análise; cálculo da matriz de

covariância, em que a matriz de covariância é calculada a partir dos dados normalizados para entender como as variáveis variam conjuntamente, ou seja, como uma variável se correlaciona com outra; cálculo dos autovalores e matriz dos componentes principais, em que a partir da matriz de covariância, calculam-se os autovalores e os autovetores, sendo que os autovalores indicam a magnitude da variação em cada direção, enquanto os autovetores (também chamados de componentes principais) indicam a direção dessas variações; seleção dos componentes principais, em que os componentes principais são ordenados de acordo com seus autovalores, e os primeiros k componentes principais são escolhidos, pois eles capturam a maior parte da variação nos dados; e transformação dos dados, em que os dados originais são projetados nos componentes principais selecionados, resultando em um conjunto de dados de dimensionalidade reduzida, mantendo as informações mais relevantes do original (JOLLIFFE, 2002).

Uma vantagem significativa da PCA é que ela facilita a visualização de dados complexos e multidimensionais, simplificando a análise sem uma perda substancial de informação. No entanto, um dos desafios é determinar o número adequado de componentes principais a serem retidos, o que frequentemente requer julgamento subjetivo ou métodos heurísticos (JOLLIFFE, 2002).

A próxima seção descreve de forma resumida a técnica de validação cruzada, que consegue dar uma estimativa mais próxima do erro real e tornar o resultado dos testes mais realista.

2.4 Validação cruzada

A Validação cruzada *k-Fold*, ou em inglês *k-Fold Cross Validation* (k-FCV), é uma técnica usada para divisão de conjuntos visando estimação de erro durante o treinamento de um modelo. A validação cruzada pode ser usada para prever melhor os erros de teste (TEMBUSAI; MAWENGKANG; ZARLIS, 2021) e tornar o resultado mais significativo e confiável (BAYKAN; YILMAZ, 2011). No *k-Fold Cross Validation*, todos os dados do conjunto de treinamento são separados aleatoriamente em k partes iguais, sendo k o número de subconjuntos. Executa-se k vezes, onde para cada vez o i-ésimo subconjunto é usado para treinamento e o restante para teste (BAYKAN; YILMAZ, 2011).

Na próxima seção serão apresentadas as métricas de avaliação, que serão utilizadas para avaliar o desempenho dos algoritmos.

2.5 Métricas de avaliação de desempenho de algoritmos

As métricas de avaliação utilizadas em tarefas de classificação são fundamentais para avaliar o desempenho de um classificador com base nos resultados obtidos (HOSSIN; SULAIMAN, 2015).

No contexto dos problemas de classificação binária, a avaliação de uma melhor solução para uma determinada classificação pode ser definida baseando-se na matriz de confusão, mostrada na Tabela 7. Por meio desta matriz, os resultados da classificação são rotulados em verdadeiro positivo (*vp*), verdadeiro negativo (*vn*), falso positivo (*fp*) e falso negativo (*fn*). O rótulo *vp* é recebido quando uma instância é positiva e foi classificada como positiva. O rótulo *vn* é recebido quando uma instância é negativa e é classificada como negativa. Ou seja, em ambos os casos os dados foram classificados corretamente. O rótulo *fp* é recebido quando uma instância é negativa e é classificada como positiva. E o rótulo *fn* é recebido quando uma instância é positiva e é classificada como negativa. Nesses dois últimos casos, os dados não foram classificados corretamente (HOSSIN; SULAIMAN, 2015).

Tabela 7 - Matriz de confusão

	Classe Positiva Real	Classe Negativa Real
Classe Positiva Prevista	Verdadeiro positivo (*vp*)	Falso positivo (*fp*)
Classe Negativa Prevista	Falso negativo (*fn*)	Verdadeiro negativo (*vn*)

Fonte: Adaptado de Hossin e Sulaiman (2015).

A partir da matriz de confusão, podem ser geradas várias métricas de avaliação, em que algumas delas são mostradas na Tabela 8 (HOSSIN; SULAIMAN, 2015).

Tabela 8 - Métricas para avaliar a classificação

Métrica	Fórmula	Foco de Avaliação
Acurácia (acc)	$\frac{vp + vn}{vp + fp + vn + fn}$	Em geral, a métrica de acurácia mede a proporção de previsões corretas sobre o total de instâncias avaliadas.
Taxa de Erro (err)	$\frac{fp + fn}{vp + fp + vn + fn}$	Mede a proporção de previsões incorretas sobre o número de instâncias avaliadas.
Sensibilidade (sn)	$\frac{vp}{vp + fn}$	Utilizada para medir a fração de padrões positivos que são classificados corretamente.
Especificidade (sp)	$\frac{vn}{vn + fp}$	Utilizada para medir a fração de padrões negativos que são classificados corretamente.
Precisão (p)	$\frac{vp}{vp + fp}$	Utilizada para medir a relação entre as previsões positivas realizadas corretamente e todas as classes positivas.
Recall (r)	$\frac{vp}{vp + vn}$	Utilizado para medir a fração de padrões positivos corretamente classificados.
F_1 *Score* (FM)	$\frac{2 * p * r}{p + r}$	Representa a média harmônica entrre os valores de precisão e *recall*.

Fonte: Adaptado de Hossin e Sulaiman (2015).

Segundo Hossin e Sulaiman (2015), a métrica de avaliação mais utilizada para problemas de classificação binária ou de múltiplas classes é a precisão. Uma métrica complementar à acurácia é a taxa de erro, que avalia a solução obtida por um classificador com base na porcentagem de previsões corretas. Uma das grandes vantagens da acurácia e da taxa de erro é a facilidade de serem calculadas, entendidas e aplicáveis a vários problemas distintos. No entanto, a acurácia e a taxa de erro possuem problema com classes desbalanceadas.

No capítulo seguinte serão apresentados os trabalhos relacionados com o projeto de mestrado.

3 LEVANTAMENTO BIBLIOGRÁFICO

Este capítulo apresenta inicialmente sobre morangos na Seção 3.1, e na Seção 3.2 apresenta alguns trabalhos relacionados para entender a metodologia proposta.

3.1 Morangos

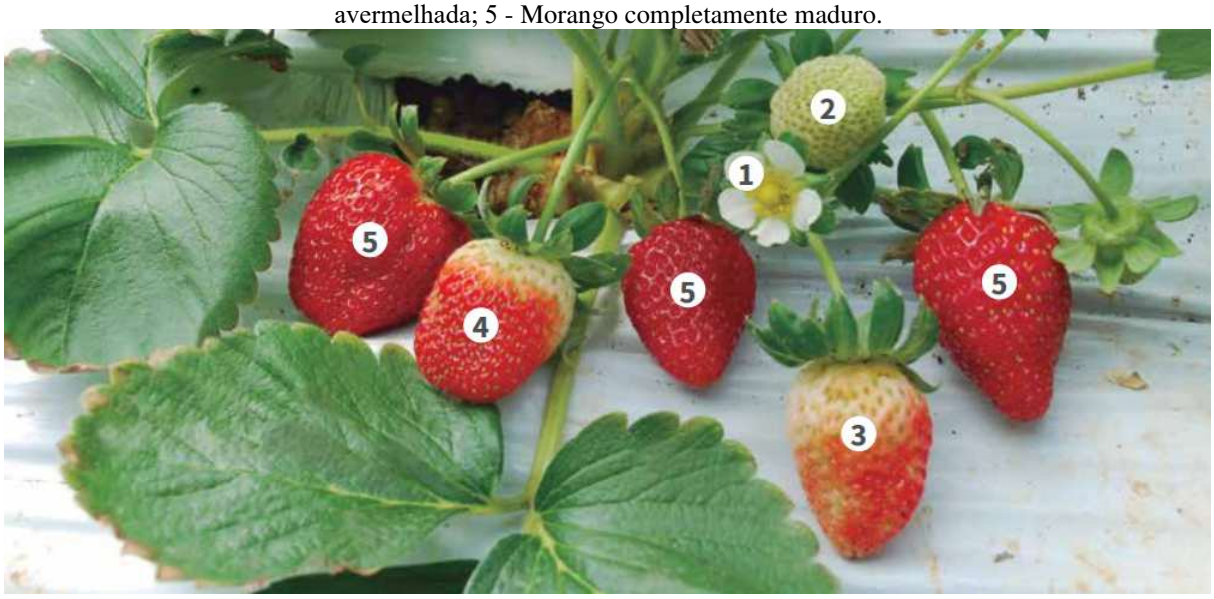
Os morangos são plantas herbáceas do tipo rasteira da família *Rosaceae* (QUINATO; DEGÁSPARI; VILELA, 2007). Eles são frutas perecíveis e não climatéricas, ou seja, eles não amadurecem depois que são colhidos (FLORES CANTILLANO; DA SILVA, 2010). A parte comestível é um pseudofruto, que cresce a partir de uma flor e se torna carnoso e suculento. Os verdadeiros frutos são as "sementes" do morango, que são chamados de aquênios. Para se desenvolverem, é necessário ocorrer a polinização, que é feita por abelhas na maior parte das vezes. Esta fruta possui propriedades medicinais. Os morangos possuem vitamina C, facilitam a digestão, aumentam a resistência às infecções, têm ação anticancerígena e são alimentos excelentes para o fígado por conta do teor elevado de açúcares naturais (QUINATO; DEGÁSPARI; VILELA, 2007).

O processo de colheita pode ocorrer em diferentes fases de maturação do morango, dependendo da finalidade para o consumo (in natura ou industrialização) (RAHMAN *et al.*, 2016). A maturação é um conjunto de mudanças físico-químicas da fruta. A coloração vermelha do morango surge por conta da presença de antocianinas, que são pigmentos naturais derivados de açúcares, e indica a maturação ideal para o consumo. Quando amadurece, as clorofilas, que possuem a coloração verde, são destruídas e as antocianinas são produzidas (FLORES CANTILLANO; DA SILVA, 2010).

A identificação do ponto de colheita é muito importante para obter morangos com boa qualidade, pois como o morango é uma fruta não climatérica, não deve ser colhido imaturo como ocorre com frutas climatéricas. Se a fruta for colhida quando estiver muito madura, pode chegar ao mercado apodrecida. E se for colhida ainda imatura, o morango ficará ácido, adstringente e sem aroma. Caso ocorra um dos casos, o valor comercial do produto será baixo. A cor é a característica mais importante para definir o ponto de colheita dos morangos. Para fins industriais, o ponto de colheita é quando o morango está com a coloração bem avermelhada, e para o consumo in natura, o ponto de colheita é quando o morango tiver, no mínimo, 70% a 75% de cor vermelha brilhante na sua superfície (BALBINO *et al.*, 2016; FLORES CANTILLANO; DA SILVA, 2010; RAHMAN *et al.*, 2016). A Figura 16 mostra os estágios de

maturação do morango. É importante ressaltar que atualmente não existe um padrão unificado para categorizar o estágio de maturação. Os agricultores ainda verificam a maturação do morango manualmente, o que é ineficiente e demorado (IBBA *et al.*, 2021).

Figura 16 - Estágios de desenvolvimento e amadurecimento do morango: 1 - Flor (Inflorescência); 2 - Verde; 3 - Morango com menos de 75% da superfície avermelhada; 4 - Morango com mais de 75% da superfície avermelhada; 5 - Morango completamente maduro.



Fonte: Balbino *et al.* (2016).

A seguir, serão apresentados os trabalhos relacionados deste projeto.

3.2 Trabalhos relacionados

Li *et al*. (2022) propuseram um modelo combinando ResNet101 e LDA para reconhecer a qualidade de aparência de morangos. A base de dados utilizada possui 4.211 imagens de morangos, e as classes foram maduro, meio maduro, não maduro e podre. O estudo utilizou seis redes neurais convolucionais (CNNs) - AlexNet, GoogleNet, ResNet34, ResNet50, ResNet101 e VGG16 - para extrair características profundas dos morangos. Essas características foram então importadas em seis classificadores diferentes, incluindo Máquina de Vetores de Suporte (SVM), Floresta Aleatória (RF), Árvores Extraordinariamente Aleatórias (ET), K-Vizinhos Mais Próximos (KNN), Perceptron Multicamadas (MLP) e Gaussian Naive Bayes (GNB), ou importadas no LDA e QDA, para prever os rótulos finais. A combinação de ResNet101 e LDA alcançou uma acurácia de 96,55%.

Herrera Perez *et al.* (2016) utilizaram a rede neural de base radial para classificar a maturação de frutos de café. A base de dados utilizada possui 248 imagens de café, sendo 70

frutos verdes e 178 frutos maduros. Foi utilizada a curva ROC para avaliar o desempenho do modelo, obtendo o resultado 0,975.

Zhang *et al.* (2018) propuseram um modelo CNN que é composta por três camadas convolucionais com ReLU, *max-pooling* e camada totalmente conectada para classificar a maturidade de bananas, e obteve uma precisão de 94,4% como resultado. A base de dados possui 17.312 imagens de bananas de diferentes estágios de maturidade.

Wan *et al.* (2018) propõem um método para detectar o nível de maturidade de tomates frescos das variedades Roma e Pera utilizando rede neural de retropropagação (BPNN). As classes foram definidas como tomate vermelho, tomate laranja (maduro) e tomate verde. A base de dados possui 150 amostras no total, com 50 imagens para cada nível de maturidade. Os resultados mostram que a precisão média para detectar os três níveis de maturidade de amostras de tomate é de 99,31%, sendo a taxa de precisão das amostras de tomate vermelho e maduro de 100,00% e a taxa de precisão para as amostras de tomate verde de 97,92%, e o desvio padrão é de 1,2%, o que pode ser utilizado para este fim.

Gao *et al.* (2020) utilizaram o modelo AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2017) pré-treinado com ImageNet (RUSSAKOVSKY *et al.*, 2015) para classificar as amostras de morango maduro e imaturo, e obteve a precisão de 98,6% para o conjunto de dados de teste. A base de dados possui 480 imagens de morangos de categorias meio maduro e maduro.

Behera, Rath e Sethy (2021) fazem uma comparação entre sete modelos pré-treinados com ImageNet (RUSSAKOVSKY *et al.*, 2015), como ResNet18, ResNet50, ResNet101 (HE *et al.*, 2016a), VGG16, VGG19 (SIMONYAN; ZISSERMAN, 2015), GoogleNet (SZEGEDY *et al.*, 2016) e AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2017) para classificar a maturidade de frutos de mamão, sendo o modelo que apresentou o melhor desempenho foi o VGG19 com 100% de precisão. A base de dados utilizada possui 300 imagens, sendo 100 para cada uma das três categorias: não-maduro, parcialmente maduro e maduro.

Psiroukis *et al.* (2022) compararam as arquiteturas Faster R-CNN com *backbone* ResNet-152, SSD com *backbone* MobileNet-V2, CenterNet com *backbone* HG-104, *RetinaNet* com *backbone* ResNet-152 e EfficientDet-D1 com *backbone* EfficientNet-D1 para a classificação de maturidade de brócolis em campo aberto e em tempo real utilizando conjunto de dados de imagens aéreas de baixa altitude. A base de dados possui um tamanho de 288 imagens e com 640 anotações. As métricas de avaliação foram baseadas em *mean Average Precision* (mAP), que é a média de *Average Precision* (AP) em todas as classes, e foram realizadas avaliações com base nas métricas que os autores chamam de mAP@50 e mAP@75,

em que são baseadas na IoU (*Intersection over Union*) maior que 0,5 e na IoU maior que 0,75, respectivamente. IoU é a área da interseção dividida pela área da união de uma caixa delimitadora prevista, e foi utilizada para decidir o que é um verdadeiro positivo para calcular a precisão. Se as previsões forem maior do que um valor definido é considerado verdadeiro positivo, caso contrário é considerado falso positivo. Se várias previsões forem maior do que o valor definido, somente a previsão com a maior IoU é considerado verdadeiro positivo e as outras são consideradas falsos positivos. Inicialmente foram definidas três classes: não-maduro (1), meio maduro (2) e maduro (3). Depois foram definidas em duas classes no conjunto de dados, onde as classes 1 e 2 foram mescladas em uma única classe e a classe 3 foi mantida. Os resultados mostraram que, em todos os experimentos, Faster R-CNN e CenterNet foram as arquiteturas de melhor desempenho para a tarefa de detecção de maturidade de brócolis. Os desempenhos dessas arquiteturas foram, em média, superiores a 80% para mAP@50 e 70% para mAP@75 no problema de três classes, independentemente da configuração de hiperparâmetros e técnicas de aumento, e ainda maior definindo para duas classes, ultrapassando 91% e 83%, respectivamente.

Cui *et al.* (2021) propõem um método de detecção com a CNN YOLOv4 aprimorado para obter a detecção em tempo real e precisa da maturidade de morango no ambiente natural. Eles utilizaram a rede de extração de características de *backbone* MobileNetv3 e convolução separável em profundidade para uma pequena melhora da rede YOLOv4. Utilizaram também o algoritmo de agrupamento Kmeans++ para calcular o tamanho da caixa delimitadora anterior para a detecção e correspondência do alvo, e usaram métodos de aprendizado de transferência e treinamento em estágios para melhorar a eficiência do treinamento do modelo. O conjunto de dados de imagem utilizado foi StrawDI_Db1, que possui dados de imagem coletados de 20 plantações de morango em diferentes ambientes de iluminação, com sobreposições, oclusões e tamanhos desiguais no crescimento, com 3100 imagens de morango. As classes foram definidas em maduro e não maduro. Os resultados experimentais mostram que a mAP do conjunto de dados de teste neste artigo é de 96,78%; a precisão de detecção de morango maduro é de 98,72%, com taxa de recall de 91,67% e precisão média (AP) de 99,56%; a precisão de detecção de morangos imaturos é de 90,76%, com taxa de recall de 83,92% e AP de 94,00%. O tempo de detecção de imagem única é de 56ms. Os resultados mostram que, apesar da precisão ser reduzida em casos de oclusão excessiva, o modelo pode detectar morangos em tempo real e com alta precisão.

Suharjito, Elwirehardja e Prayoga (2021) realizaram uma comparação entre os modelos EfficientNetB0, NASNet, MobileNetV1 e MobileNetV2, utilizando os pesos de ImageNet

(RUSSAKOVSKY *et al.*, 2015), para classificar os níveis de amadurecimento do óleo de palma. A base de dados possui 653 imagens de óleo de palma de seis categorias: não-maduro, pouco maduro, maduro, maduro demais, anormal e cacho de frutas vazio. O modelo com melhor resultado foi o EfficientNetB0 com uma precisão geral de teste de 0,893 e o segundo melhor modelo foi MobileNetV1 com precisão de 0,811.

Khosravi, Saedi e Rezaei (2021) propuseram um modelo CNN com três camadas convolucionais, *max-pooling*, e *Global Average Pooling* (GAP) com *Batch Normalization*, para classificar a maturidade de azeitonas iranianas, e a acurácia geral do modelo proposto foi de 91,91%. A base de dados utilizada possui 14.895 imagens de azeitonas das classes não-maduro, maduro verde, maduro preto e totalmente maduro.

Hendrawan *et al.* (2021) comparam quatro tipos de CNNs pré-treinadas (SqueezeNet, GoogLeNet, ResNet50 e AlexNet) para classificar o estágio de maturidade da pimenta verde grande em três níveis de maturidade, como pimentas colhidas aos 34 dias após a abertura de botões de flor (DAA), pimentas colhidas aos 47 DAA e pimentas colhidas aos 60 DAA. A base de dados possui 1200 imagens. Os resultados mostraram que os quatro modelos CNN pré-treinados das redes produziram diferentes acurácias de classificação com uma acurácia variando desde 82,22% até 93,89%.

Benmouna *et al.* (2022) propuseram uma CNN que emprega convoluções 1-D em suas camadas convolucionais para classificar a maturidade da maçã Fuji. A base de dados possui 172 imagens para quatro classes: não-maduro, meio maduro, maduro e maduro demais. A CNN possui duas camadas convolucionais com 64 *kernels*, uma camada de *max-pooling*, uma camada *Flatten* e duas camadas densas (ou totalmente conectadas) com 100 unidades, e obteve uma precisão média de aproximadamente 95,59%.

Em todos os trabalhos relatados acima foram utilizados conjunto de dados próprio, e que não foram publicados. Esse levantamento de trabalhos relacionados mostra que as CNNs são frequentemente utilizadas na área de classificação de maturidade de frutas em geral.

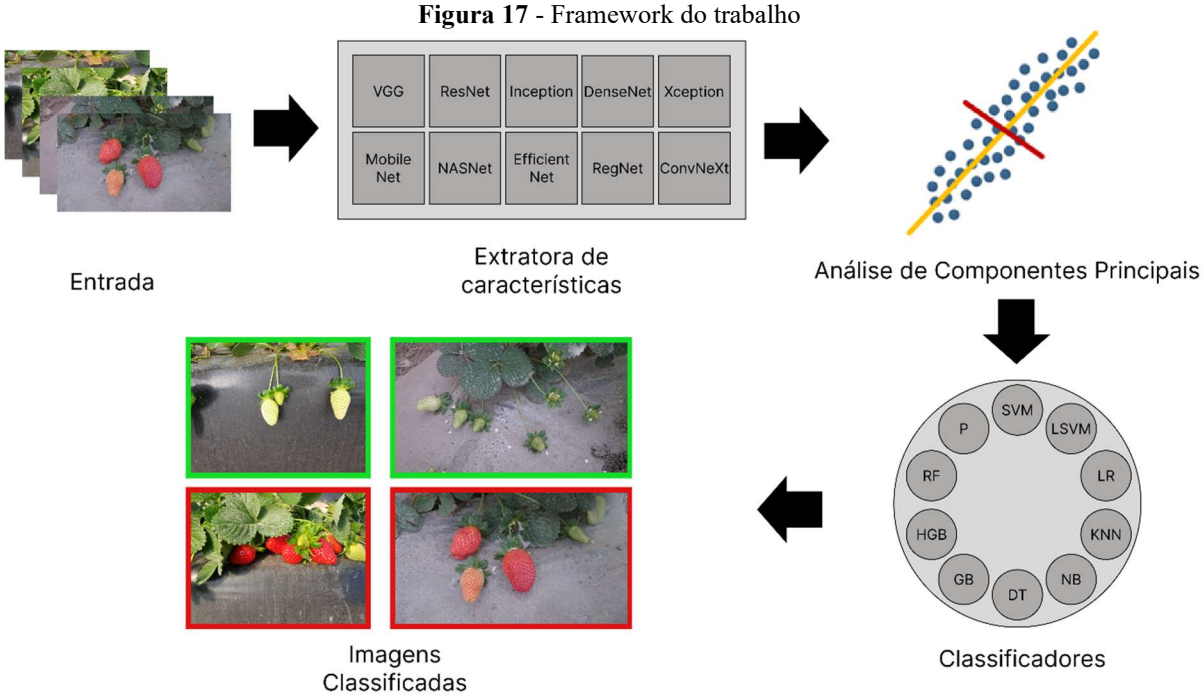
Estes trabalhos foram encontrados por meio de ScienceDirect (filtrando por “Computer Science” em “Subject areas”), Scielo e Springer Link (com “Include Preview-Only content” desselecionado), pesquisando por “Strawberry Maturity Classification”, “Strawberry Ripeness Classification”, “Strawberry Maturity CNN”, “Strawberry Ripeness CNN”, “Fruit Maturity Classification”, “Fruit Ripeness Classification”, “Fruit Maturity CNN” e “Fruit Ripeness CNN”, “Deep Feature Strawberry Maturity Classification”.

No capítulo seguinte será apresentada a metodologia utilizada neste trabalho.

4 METODOLOGIA

Neste Capítulo é apresentada uma descrição do método proposto para classificação de imagens de morangos. A Figura 17 mostra o processo do trabalho de forma visual.

Figura 17 - Framework do trabalho

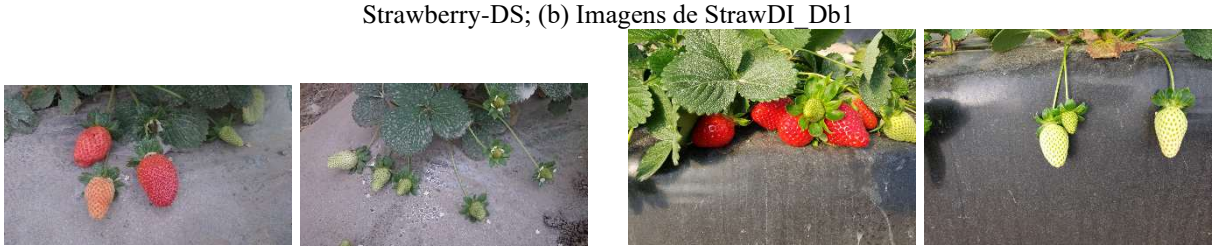


Fonte: A autora (2024).

Neste trabalho, a ordem dos processos são a extração de características das imagens com as extratoras usando ImageNet (RUSSAKOVSKY *et al.*, 2015), separar os índices das imagens em *K-Folds* para fazer o *Cross Validation* e realizar a classificação com base nas características extraídas, estas sendo passadas por PCA. Nesse caso, o PCA foi necessário, pois a saída de alguns modelos era bem grande, e a memória RAM disponível não era suficiente para carregar tudo. Os testes foram realizados sem balanceamento de dados e com balanceamento de dados. O balanceamento de dados foi realizado com o ajuste de pesos dos classificadores de forma automática pelo Sklearn (SCIKIT-LEARN, 2024), quando declara '*balanced*' como parâmetro de *class_weight*. Se não for fornecido nenhum parâmetro, todas as classes deverão ter peso um. O modo "balanceado" usa os valores de classes para ajustar automaticamente os pesos inversamente proporcionais às frequências de classe nos dados de entrada.

Os conjuntos de dados de imagem utilizados neste trabalho foram o Strawberry-DS[1] e o StrawDI_Db1[2]. O Strawberry-DS possui 247 imagens em RGB de plantações de morango com resolução de 3840x2160 pixels (ELHARIRI; EL-BENDARY; SALEH, 2023). E o StrawDI_Db1 possui 3100 imagens de plantações de morango com resolução de 1008x756 pixels (CUI *et al.*, 2021). As imagens foram separadas e rotuladas como “Colher” e “Nao_colher”, sendo “Colher” as imagens que têm pelo menos um morango para colher, e “Nao_colher” as imagens sem morangos prontos para colher. No total, o número de imagens do conjunto Strawberry-DS na classe “Colher” são 170 (equivalente a aproximadamente 68,83% da base de dados) e na classe “Nao_colher” são 77 (equivalente a aproximadamente 31,17% da base de dados), e do conjunto StrawDI_Db1 na classe “Colher” são 1780 (equivalente a aproximadamente 57,42% da base de dados) e na classe “Nao_colher” são 1320 (equivalente a aproximadamente 42,58% da base de dados). Alguns exemplos de imagens de ambos os conjuntos de dados estão sendo mostrados na Figura 18.

Figura 18 - Exemplos de imagens dos conjuntos de dados Strawberry-DS e StrawDI_Db1: (a) Imagens de Strawberry-DS; (b) Imagens de StrawDI_Db1



(a) (b)

Fonte: Elhariri, El-Bendary e Saleh (2023) e Pérez-Borrero *et al.* (2020).

O quadro 1 mostra a relação de CNNs que foram usadas neste trabalho. Foram escolhidas estas CNNs, pois foram todos os modelos que estavam disponíveis na plataforma Tensorflow (ABADI *et al.*, 2015), na versão 2.15.1 que estava quando o projeto foi iniciado.

[1]Pode ser baixado em https://data.mendeley.com/datasets/z6dtfdpzz8/1
[2]Pode ser baixado em https://strawdi.github.io/

Quadro 1 - Relação de CNNs

Modelo	**Referência**
VGG16, VGG19	SIMONYAN; ZISSERMAN, 2015
ResNet50, ResNet101, ResNet152	HE *et al*., 2016a
ResNet50V2, ResNet101V2, ResNet152V2	HE *et al*., 2016b
ResNetRS50, ResNetRS101, ResNetRS152, ResNetRS200, ResNetRS270, ResNetRS350, ResNetRS420	BELLO *et al*., 2021
InceptionV3	SZEGEDY *et al*., 2016
InceptionResNetV2	SZEGEDY *et al*., 2017
DenseNet121, DenseNet169, DenseNet201	HUANG *et al*., 2017
Xception	CHOLLET, 2017
MobileNet	HOWARD *et al*., 2017
MobileNetV2	SANDLER *et al*., 2018
MobileNetV3Large, MobileNetV3Small	HOWARD *et al*., 2019
NASNetLarge, NASNetMobile	ZOPH *et al*., 2018
EfficientNetB0, EfficientNetB1, EfficientNetB2, EfficientNetB3, EfficientNetB4, EfficientNetB5, EfficientNetB6, EfficientNetB7	TAN; LE, 2019
EfficientNetV2B0, EfficientNetV2B1, EfficientNetV2B2, EfficientNetV2B3, EfficientNetV2S, EfficientNetV2M, EfficientNetV2L	TAN; LE, 2021
RegNetX002, RegNetX004, RegNetX006, RegNetX008, RegNetX016, RegNetX032, RegNetX040, RegNetX064, RegNetX080, RegNetX120, RegNetX160, RegNetX320, RegNetY002, RegNetY004, RegNetY006, RegNetY008, RegNetY016, RegNetY032, RegNetY040, RegNetY064, RegNetY080, RegNetY120, RegNetY160, RegNetY320	RADOSAVOVIC *et al.*, 2020
ConvNeXtTiny, ConvNeXtSmall, ConvNeXtBase, ConvNeXtLarge, ConvNeXtXLarge	LIU *et al*., 2022

Fonte: A autora (2024).

Essas redes convolucionais foram utilizadas como extratoras de características. Elas são usadas sem a última camada, a camada de classificação, com pesos pré-treinados no conjunto de dados ImageNet (RUSSAKOVSKY *et al.*, 2015). O tamanho das imagens foi definido conforme a documentação da plataforma Tensorflow (ABADI *et al.*), sendo em geral com 224x224 pixels, exceto InceptionV3, InceptionResNetV2, Xception (com 299x299 pixels), NASNetLarge e NASNetMobile (com 331x331 pixels), foram utilizados os três canais RGB e não foi utilizado o *pooling*. Os resultados da extração foram salvos em um arquivo CSV para cada extratora.

Após a extração, os valores resultantes foram normalizados com StandardScaler de Sklearn (SCIKIT-LEARN, 2024), depois foram divididos em 5 *folds* e passados para o PCA, testando entre 90% a 99% de variância retida para reduzir a dimensionalidade de dados, e depois foram passados para os modelos SVM, SVM Linear (LSVM), regressão logística (LR), KNN, *Naive Bayes* (NB), árvore de decisão (DT), *Gradient Boosting* (GB), *Histogram-based Gradient Boosting* (HGB), *Random Forest* (RF) e Perceptron (P) para realizar a classificação, com e sem balanceamento de pesos (exceto para KNN, Naive Bayes e *Gradient Boosting*, pois não havia parâmetro no Sklearn (SCIKIT-LEARN, 2024) para realizar o balanceamento de pesos). Para SVM, SVM Linear e regressão logística foi definido o número de iterações máximo para 1.000.000 iterações.

Para avaliar os resultados obtidos, foram utilizadas as métricas de acurácia, F_1 *Score*, *Recall* e precisão. Mas para a questão de comparação, foi usada a F_1 *Score*, pois utilizar somente a acurácia não seria o suficiente por envolver um certo desbalanceamento de dados.

Os resultados obtidos serão mostrados no capítulo a seguir.

5 RESULTADOS

A plataforma Google Colab (GOOGLE, 2024) versão gratuita foi utilizada na programação dos modelos. Ela utiliza como linguagem de programação o Python versão 3 (PYTHON, 2024) e executa os códigos em nuvem do Google, ou seja, a plataforma fornece os recursos computacionais para executar códigos sem ter a necessidade de haver recursos computacionais potentes na própria máquina. A memória RAM é de 12,67GB fixo, mas a configuração do hardware é definida a cada vez que o ambiente de execução é conectado, e como pode ocasionalmente desconectar e reconectar o ambiente, os tempos de execução não foram trazidos como resultado por deixar o tempo de execução bastante impreciso.

Foram executadas todas as combinações de extratoras e classificadores. No conjunto de dados Strawberry-DS foi possível obter os resultados de todas as combinações. Porém, ao executar a extração de características com os modelos Xception, NASNetLarge, RegNetY160 e RegNetY320 no conjunto de dados StrawDI_Db1, a memória RAM foi insuficiente para concluir a extração. Além disso, a memória RAM foi insuficiente também ao carregar os arquivos CSV dos modelos InceptionV3, DenseNet201, ResNetRS152, ResNetRS200, ResNetRS270, ResNetRS350, ResNetRS420, EfficientNetB0, EfficientNetB1, EfficientNetB2, EfficientNetB3, EfficientNetB4, EfficientNetB5, EfficientNetB6, EfficientNetB7, EfficientNetV2B0, EfficientNetV2B1, EfficientNetV2B2, EfficientNetV2B3, EfficientNetV2S, EfficientNetV2M, EfficientNetV2L, ConvNeXtLarge e ConvNeXtXLarge para poder realizar a classificação.

A Tabela 9 apresenta a média dos *k-Folds* dos dez melhores resultados obtidos em cada um dos modelos testados com Strawberry-DS.

Tabela 9 – Resultados obtidos com combinações de modelos usando 5 *k-Folds* com Strawberry-DS

Técnicas Aplicadas	Dados balanceados	Variância retida de PCA	Número de atributos	Acurácia	F1-Score	Recall	Precisão
VGG16 + GB	Não	0,97	218	0,77959	0,85119	0,91176	0,79909
VGG19 + P	Não	0,96	209	0,75918	0,85087	0,98235	0,75172
VGG19 + HGB	Sim	0,9	175	0,78367	0,84684	0,84559	0,82928
VGG19 + HGB	Não	0,9	175	0,77959	0,84672	0,88824	0,81203
ConvNeXtTiny + LR	Não	0,9	85	0,77959	0,84487	0,87059	0,82264
VGG19 + GB	Não	0,91	180	0,77143	0,84477	0,90588	0,79316
VGG19 + LR	Não	0,93	190	0,75918	0,84244	0,92941	0,77044
ConvNeXtBase + SVM	Sim	0,94	156	0,73878	0,83849	0,97059	0,73500
VGG19 + LR	Sim	0,95	202	0,75102	0,83673	0,91176	0,76971
VGG19 + RF	Sim	0,9	175	0,72653	0,83582	1,00000	0,71844

Técnicas Aplicadas	Dados balanceados	Variância retida de PCA	Desvio padrão de acurácia	Desvio padrão de F1-Score	Desvio padrão de Recall	Desvio padrão de Precisão
VGG16 + GB	Não	0,97	0,03957	0,02896	0,04922	0,02204
VGG19 + P	Não	0,96	0,05685	0,03105	0,01441	0,04760
VGG19 + HGB	Sim	0,9	0,04582	0,03745	0,07804	0,03890
VGG19 + HGB	Não	0,9	0,07570	0,05728	0,08998	0,04405
ConvNeXtTiny + LR	Não	0,9	0,08792	0,06314	0,08235	0,05759
VGG19 + GB	Não	0,91	0,05227	0,04087	0,07533	0,01779
VGG19 + LR	Não	0,93	0,04358	0,02991	0,03990	0,02281
ConvNeXtBase + SVM	Sim	0,94	0,02999	0,01734	0,02201	0,02076
VGG19 + LR	Sim	0,95	0,06244	0,03914	0,04323	0,04332
VGG19 + RF	Sim	0,9	0,03785	0,01948	0,00000	0,02926

Fonte: A autora (2024).

A Tabela 10 apresenta a média dos *k-Folds* dos dez melhores resultados obtidos em cada um dos modelos testados com StrawDI_Db1.

Tabela 10 – Resultados obtidos com combinações de modelos usando 5 *k-Folds* com StrawDI_Db1

Técnicas Aplicadas	Dados balanc.	Variância retida de PCA	Número de atributos	Acurácia	F1-Score	Recall	Precisão
ConvNeXtBase + GB	Não	0,91	1367	0,76935	0,81105	0,86124	0,76661
ConvNeXtBase + HGB	Não	0,93	1572	0,76774	0,81052	0,86461	0,76298
ConvNeXtBase + HGB	Sim	0,91	1367	0,76645	0,80672	0,84831	0,76914
ConvNeXtBase + SVM	Não	0,9	1278	0,72129	0,79648	0,94944	0,68606
ConvNeXtSmall + GB	Não	0,95	1678	0,74710	0,79214	0,83933	0,75007
ConvNeXtSmall + HGB	Não	0,91	1235	0,74419	0,79159	0,84551	0,74428
ConvNeXtSmall + SVM	Não	0,9	1152	0,72226	0,78780	0,89775	0,70187
ConvNeXtSmall + HGB	Sim	0,92	1327	0,74581	0,78685	0,81685	0,75927
ConvNeXtTiny + HGB	Não	0,91	1512	0,73194	0,78465	0,85000	0,72909
ConvNeXtSmall + SVM	Sim	0,9	1152	0,73258	0,78178	0,83371	0,73605

Técnicas Aplicadas	Dados balanc.	Variância retida de PCA	Desvio padrão de acurácia	Desvio padrão de F1-Score	Desvio padrão de Recall	Desvio padrão de Precisão
ConvNeXtBase + GB	Não	0,91	0,01813	0,01275	0,00550	0,02071
ConvNeXtBase + HGB	Não	0,93	0,01290	0,00835	0,00373	0,01560
ConvNeXtBase + HGB	Sim	0,91	0,01717	0,01274	0,00795	0,01813
ConvNeXtBase + SVM	Não	0,9	0,01222	0,00707	0,00533	0,01082
ConvNeXtSmall + GB	Não	0,95	0,02340	0,01932	0,02229	0,01888
ConvNeXtSmall + HGB	Não	0,91	0,02208	0,01653	0,01388	0,02060
ConvNeXtSmall + SVM	Não	0,9	0,01794	0,01332	0,01450	0,01312
ConvNeXtSmall + HGB	Sim	0,92	0,01375	0,00999	0,01236	0,01655
ConvNeXtTiny + HGB	Não	0,91	0,03056	0,02354	0,02698	0,02800
ConvNeXtSmall + SVM	Sim	0,9	0,02023	0,01488	0,01027	0,01935

Fonte: A autora (2024).

Com base nos resultados obtidos, pode-se observar que no conjunto Strawberry-DS os modelos mais adequados para a tarefa de classificação do grau de maturidade de morangos

foram da família VGG, que é um modelo relativamente antigo, com os classificadores *Gradient Boosting*, Perceptron, *Histogram Based Gradient Boosting*, regressão logística, SVM e *Random Forest*, e no conjunto StrawDI_Db1 foram da família ConvNeXt, que é um modelo mais recente, com os classificadores *Gradient Boosting*, *Histogram Based Gradient Boosting* e SVM, e em ambos os casos foi melhor sem o balanceamento de pesos. Em geral, a acurácia foi acima de 72% e F_1 *Score* foi acima de 78%. As extratoras de características mais adequadas para esta tarefa, em ambos os casos, são da família ConvNeXt e os classificadores supervisionados mais adequados são *Gradient Boosting*, *Histogram Based Gradient Boosting* e SVM, para esta tarefa. O código[3] e o resultado[4] completo com informações extras podem ser acessado pelos links no rodapé.

[3]Acesso ao código do projeto: https://colab.research.google.com/drive/1IRLMAMZ-_ICvycY4Z6QSWOaW1076lWGM?usp=sharing

[4]Acesso ao resultado completo do projeto: https://docs.google.com/spreadsheets/d/1Kav5MiMjB_3QH7xr-oDeiOPC1h4lTToO/edit?usp=sharing&ouid=109675010882375249930&rtpof=true&sd=true

6 CONCLUSÃO

Neste trabalho, foram estudados os morangos e o seu momento de colheita, o aprendizado de máquina e uma de suas principais categorias, o aprendizado supervisionado, alguns conceitos de algoritmos como redes neurais artificiais, redes neurais convolucionais, modelos supervisionados, redução de dimensionalidade, validação cruzada e métricas de avaliação.

Em seguida, foi apresentada a metodologia que foi utilizada no projeto de mestrado e apresentados os resultados obtidos com as combinações. Os melhores resultados obtidos foram com as CNNs da família ConvNeXt (ConvNeXtBase, ConvNeXtSmall e ConvNeXtTiny) e da família VGG (VGG16 e VGG19), combinando com os classificadores *Gradient Boosting*, *Histogram Based Gradient Boosting* e SVM, alcançando acurácia acima de 72% e F1-Score acima de 78%. Com base nisso, pode-se afirmar que é possível realizar de forma eficiente e precisa a tarefa de classificação do grau de maturidade de morangos com CNNs como extratoras de características e classificadores supervisionados, podendo beneficiar os agricultores com uma ferramenta mais eficaz e menos custosa.

Os problemas enfrentados no projeto foram a limitação dos recursos computacionais e o tempo escasso para realizar os testes. O Google Colab fornece um ambiente de execução de forma gratuita por até no máximo 12h consecutivas. Se passar desse tempo, o ambiente de execução é desconectado, perdendo os treinamentos realizados nos modelos caso não tenham sido salvos. Apesar de fornecerem 12,7 GB de memória RAM, ainda não foi suficiente para alguns modelos em que estavam sendo trabalhados.

Como trabalho futuro, pretende-se publicar um artigo deste projeto na área de aplicação. Além disso, pretende-se estudar algoritmos para segmentação de imagens para inserir no projeto, combinar as features, juntar as bases para treinamento, utilizar outros algoritmos de redução de dimensionalidade e de seleção de melhores características, como Isomap, LDA, Relief e ReliefF, implementar técnicas de IA explicável, como CAM, Grad-CAM e LIME, para saber quantos morangos maduros influenciam na saída, e, por fim, criar e implementar uma aplicação real para celulares ou mesmo para drones. Caso consiga recursos computacionais mais robustos, pretende-se treinar as CNNs e também testar algoritmos para detecção de morangos maduros.

REFERÊNCIAS

ABADI, M. *et al.* TensorFlow: large-scale machine learning on heterogeneous systems. **Preliminary White Paper**, [S. l.], p. 1-19, 2015. Disponível em: <https://www.tensorflow.org/>. Acesso em: 30 jul. 2024.

AHMADI, S. *et al*. The Strategic Perceptron. **Proceedings of The 22nd Acm Conference on Economics and Computation**, [S. l.], p. 6-25, 18 jul. 2021. Disponível em: <https://doi.org/10.1145/3465456.3467629>. Acesso em: 22 jan. 2024.

ALPAYDIN, E. **Introduction to machine learning.** 2. ed. Massachusetts: MIT Press, 2010. 539p. Disponível em: <https://ds.amu.edu.et/xmlui/bitstream/handle/123456789/6943/Introduction%20to%20Machine%20Learning.pdf?sequence=1&isAllowed=y>. Acesso em: 01 abr. 2022.

ALZUBAIDI, L. *et al*. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. **Journal of Big Data**, [S. l.], v. 8, n. 53, p. 1-74, 2021. Disponível em: <https://doi.org/10.1186/s40537-021-00444-8>. Acesso em: 01 jul. 2022.

ANDREW, A.; SANTOSO, H. Compare VGG19, ResNet50, Inception-V3 for review food rating. **Sinkron**: Jurnal Dan Penelitian Teknik Informatika, [S. l.], v. 7, n. 2, p. 845-494, 2022. Disponível em: <https://doi.org/10.33395/sinkron.v7i2.11383>. Acesso em: 31 dez. 2022.

AWAD, M.; KHANNA, R. Support Vector Machines for Classification. **Efficient Learning Machines**, [S. l.], p. 39-66, 2015. Disponível em: <https://doi.org/10.1007/978-1-4302-5990-9_3>. Acesso em: 22 jan. 2024.

BALBINO, J. M. S. *et al*. **Boas práticas de colheita e pós-colheita**: qualidade e aproveitamento do morango. Vitória: Incaper, 2016. 23p. (Incaper. Documentos, 241). Disponível em: <https://biblioteca.incaper.es.gov.br/digital/bitstream/item/2083/1/BRT-boaspraticasmorango-Incaper.pdf>. Acesso em: 15 jan. 2023.

BAYKAN, N.; YILMAZ, N. A mineral classification system with multiple artificial neural network using k-fold cross validation. **Mathematical and Computational Applications**, Konya, v. 16, n. 1, p. 22-30, 2011. Disponível em: <https://doi.org/10.3390/mca16010022>. Acesso em: 31 dez. 2022.

BEHERA, S. K.; RATH, A. K.; SETHY, P. K. Maturity status classification of papaya fruits based on machine learning and transfer learning approach. **Information Processing in Agriculture**, [S. l.], v. 8, n. 2, p. 244-250, 2021. Disponível em: <https://doi.org/10.1016/j.inpa.2020.05.003>. Acesso em: 31 dez. 2022.

BELLO, I. *et al*. Revisiting ResNets: Improved Training and Scaling Strategies. In: RANZATO, M.; BEYGELZIMER, A.; DAUPHIN, Y.; LIANG, P. S.; WORTMAN VAUGHAN, J. (Ed.). **Advances in Neural Information Processing Systems**, v. 34, p. 22614-22627, 2021. Curran Associates, Inc. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2021/file/bef4d169d8bddd17d68303877a3ea945-Paper.pdf>. Acesso em: 22 jan. 2024.

BENDJILLALI, R. *et al*. Illumination-robust face recognition based on deep convolutional neural networks architectures. **Indonesian Journal of Electrical Engineering and Computer Science**, [S. l.], v. 18, n. 2, p. 1015-1027, 2020. Disponível em: <https://doi.org/10.11591/ijeecs.v18.i2.pp1015-1027>. Acesso em: 31 dez. 2022.

BENMOUNA, B. *et al*. Convolutional neural networks for estimating the ripening state of Fuji apples using visible and near-infrared spectroscopy. **Food and Bioprocess Technology**, [S. l.], v. 15, n. 10, p. 2226-2236, 2022. Disponível em: <https://doi.org/10.1007/s11947-022-02880-7>. Acesso em: 31 dez. 2022.

BEVILACQUA, H. E. C. R. Classificação das hortaliças. In: CASTANHEIRO, ALM; BEVILACQUA, HECR; SHIRAKI, JN (Coords.). **Horta**: cultivo de hortaliças. São Paulo: Prefeitura do Município de São Paulo, Secretaria Municipal do Verde e do Meio Ambiente, 2006. p. 1039-1042. Disponível em: <https://www.agriculturaurbana.org.br/textos/manual_horta.pdf>. Acesso em: 01 abr. 2022.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. 1. ed. New York: Springer, 2006. (Information Science and Statistics). Disponível em: <https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf>. Acesso em: 22 jan. 2024. ISBN 978-0-387-31073-2.

BLUM, A. **Machine learning theory**. Pittsburgh: Carnegie Mellon University, 2007. v. 26, 4p. Disponível em: <https://www.cs.cmu.edu/afs/cs/user/avrim/www/Talks/mlt.pdf>. Acesso em: 01 jul. 2022.

CHOLLET, F. Xception: deep learning with depthwise separable convolutions. **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, [S. l.], p. 1800-1807, jul. 2017. Disponível em: <https://doi.org/10.1109/CVPR.2017.195>. Acesso em: 22 jan. 2024.

CUI, M. *et al*. Research on Strawberry Maturity Detection Technology Based on Improved YOLOv4. **Journal of Physics**: Conference Series, [S. l.], v. 2138. n. 1, p. 012012, 2021. Disponível em: <https://doi.org/10.1088/1742-6596/2138/1/012012>. Acesso em: 01 jul. 2022.

DEBIE, E.; SHAFI, K. Implications of the curse of dimensionality for supervised learning classifier systems: theoretical and empirical analyses. **Pattern Analysis and Applications**, [S. l.], v. 22, n. 2, p. 519-536, 2017. Disponível em: <https://doi.org/10.1007/s10044-017-0649-0>. Acesso em: 31 dez. 2022.

DENG, L.; YU, D. Deep Learning: methods and applications. **Foundations and Trends® in Signal Processing**, Boston, v. 7, n. 3-4, p. 197-387, 2014. Now Publishers. Disponível em: <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/DeepLearning-NowPublishing-Vol7-SIG-039.pdf>. Acesso em: 31 dez. 2022.

ELHARIRI, E.; EL-BENDARY, N.; SALEH, S. M.. Strawberry-DS: dataset of annotated strawberry fruits images with various developmental stages. **Data in Brief**, [S. l.], v. 48, p. 109165, jun. 2023. Disponível em: <https://doi.org/10.1016/j.dib.2023.109165>. Acesso em: 01 jul. 2023.

FISHER, E. M. Linear Discriminant Analysis. **Statistics & Discrete Methods of Data Sciences**, [S. l.], p. 1-5, 1936.

FLORES CANTILLANO, R. F.; DA SILVA, M. M. **Manuseio pós-colheita de morangos**. Pelotas: Embrapa Clima Temperado, 2010. 36p. (Embrapa Clima Temperado. Documentos, 318). Disponível em: <https://ainfo.cnptia.embrapa.br/digital/bitstream/item/44009/1/documento-318.pdf>. Acesso em: 15 jan. 2023.

FRIEDMAN, J. H. Greedy function approximation: a gradient boosting machine. **The Annals of Statistics**, [S. l.], v. 29, n. 5, p. 1189-1232, 1 out. 2001. Disponível em: <https://doi.org/10.1214/aos/1013203451>. Acesso em: 22 jan. 2024.

FUKUSHIMA, K. Self-organization of a neural network which gives position-invariant response. **Proceedings International Joint Conference on Artificial intelligence**, [S. l.], v. 6, n. 1, p. 291-293, 1979.

GALANIS, N. I. *et al.* Convolutional Neural Networks: a roundup and benchmark of their pooling layer variants. **Algorithms**, [S. l.], v. 15, n. 11, p. 391, 23 out. 2022. Disponível em: <https://doi.org/10.3390/a15110391>. Acesso em: 22 jan. 2024.

GAO, Z. *et al*. Real-time hyperspectral imaging for the in-field estimation of strawberry ripeness with deep learning. **Artificial Intelligence in Agriculture**, [S. l.], v. 4, p. 31-38, 2020. Disponível em: <https://doi.org/10.1016/j.aiia.2020.04.003>. Acesso em: 31 dez. 2022.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S. l.]: MIT Press, 2016. 781p. Disponível em: <http://www.deeplearningbook.org>. Acesso em: 01 jul. 2022.

GOODFELLOW, I. *et al.* Generative adversarial networks. **Communications of the ACM**, [S. l.], v. 63, n. 11, p. 139-144, 2020. Disponível em: <https://doi.org/10.1145/3422622>. Acesso em: 31 dez. 2022.

GOOGLE. **Google Colaboratory**. Disponível em: <https://colab.google/>. Acesso em: 07 fev. 2024.

HAKIM, H.; FADHIL, A. Survey: convolution neural networks in object detection. **Journal of Physics**: Conference Series, [S. l.], v. 1804, n. 1, p. 012095, 2021. Disponível em: <https://doi.org/10.1088/1742-6596/1804/1/012095>. Acesso em: 01 abr. 2022.

HAYKIN, S. **Neural Networks**: A Comprehensive Foundation. 2. ed. Englewood Cliffs: Prentice-Hall, 1999. 823p. Disponível em: <https://cdn.preterhuman.net/texts/science_and_technology/artificial_intelligence/Neural%20Networks%20-%20A%20Comprehensive%20Foundation%20-%20Simon%20Haykin.pdf>. Acesso em: 31 dez. 2022.

HE, K. *et al.* Deep residual learning for image recognition. **2016 IEEE Conference on Computer Vision and Pattern Recognition**, [S. l.], p. 770-778, 2016a. Disponível em: <https://doi.org/10.1109/CVPR.2016.90>. Acesso em: 31 dez. 2022.

HE, K. *et al*. Identity Mappings in Deep Residual Networks. **Computer Vision – Eccv 2016**, [S. l.], p. 630-645, 2016b. Disponível em: <https://doi.org/10.1007/978-3-319-46493-0_38>. Acesso em: 22 jan. 2024.

HENDRAWAN, Y. *et al*. Classification of large green chilli maturity using deep learning**. Iop Conference Series**: Earth and Environmental Science, [S. l.], v. 924, n. 1, p. 012009, 1 nov. 2021. Disponível em: <https://doi.org/10.1088/1755-1315/924/1/012009>. Acesso em: 01 abr. 2022.

HERRERA PEREZ, J. C. *et al*. Clasificación de los frutos de café según su estado de maduración y detección de la broca mediante técnicas de procesamiento de imágenes. **Prospectiva**, Barranquilla, v. 14, n. 1, p. 15-22, 2016. Disponível em: <https://doi.org/10.15665/rp.v14i1.640>. Acesso em: 31 dez. 2022.

HOSSIN, M.; SULAIMAN, M. N. A review on evaluation metrics for data classification evaluations. **International Journal of Data Mining & Knowledge Management Process**, [S. l.], v. 5, n. 2, p. 1-11, 2015. Disponível em: <https://doi.org/10.5121/ijdkp.2015.5201>. Acesso em: 31 dez. 2022.

HOWARD, A. G. *et al*. MobileNets: efficient convolutional neural networks for mobile vision applications. **arXiv**, [S. l.], p. 1-9, 2017. Disponível em: <https://doi.org/10.48550/arXiv.1704.04861>. Acesso em: 22 jan. 2024.

HOWARD, A. G. *et al*. Searching for MobileNetV3. **2019 IEEE/CVF International Conference on Computer Vision (ICCV)**, [S. l.], p. 1314-1324, out. 2019. Disponível em: <https://doi.org/10.1109/ICCV.2019.00140>. Acesso em: 22 jan. 2024.

HUANG, G. *et al*. Densely Connected Convolutional Networks. **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, [S. l.], p. 2261-2269, jul. 2017. Disponível em: <https://doi.org/10.1109/CVPR.2017.243>. Acesso em: 22 jan. 2024.

IBBA, P. *et al*. Supervised binary classification methods for strawberry ripeness discrimination from bioimpedance data. **Scientific Reports**, [S. l.], v. 11, n. 1, p. 1-13, 27 maio 2021. Disponível em: <https://doi.org/10.1038/s41598-021-90471-5>. Acesso em: 22 jan. 2024.

JOLLIFFE, I. Principal Component Analysis. **Springer Series In Statistics**, [S. l.], p. 1-488, 2002. Disponível em: <https://doi.org/10.1007/b98835>. Acesso em: 01 jan. 2023.

KHOSRAVI, H.; SAEDI, S. I.; REZAEI, M. Real-time recognition of on-branch olive ripening stages by a deep convolutional neural network. **Scientia Horticulturae**, [S. l.], v. 287, p. 110252, 2021. Disponível em: <https://doi.org/10.1016/j.scienta.2021.110252>. Acesso em: 31 dez. 2022.

KONONENKO, I. Estimating attributes: analysis and extensions of RELIEF. **European conference on machine learning**, [S. l.], p. 171-182, 1994. Disponível em: <https://doi.org/10.1007/3-540-57868-4_57>. Acesso em: 31 dez. 2022.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Communications of the ACM**, [S. l.], v. 60, n. 6, p. 84-90, 2017. Disponível em: <https://doi.org/10.1145/3065386>. Acesso em: 31 dez. 2022.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, [S. l.], v. 521, n. 7553, p. 436-444, 27 maio 2015. Disponível em: <https://doi.org/10.1038/nature14539>. Acesso em: 31 dez. 2022.

LECUN, Y. *et al.* Object Recognition with Gradient-Based Learning. **Shape, Contour and Grouping in Computer Vision**, [S. l.], p. 319-345, 1999. Disponível em: <https://doi.org/10.1007/3-540-46805-6_19>. Acesso em: 01 jan. 2023.

LI, X. *et al.* Recognizing strawberry appearance quality using different combinations of deep feature and classifiers. **Journal of Food Process Engineering**, [S. l.], v. 45, n. 3, p. 1-11, 17 fev. 2022. Disponível em: <https://doi.org/10.1111/jfpe.13982>. Acesso em: 22 jan. 2024.

LIU, Z. *et al*. A ConvNet for the 2020s. **2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**, [S. l.], p. 11966-11976, jun. 2022. Disponível em: <https://doi.org/10.1109/CVPR52688.2022.01167>. Acesso em: 22 jan. 2024.

MITCHELL, T. M. **Machine learning.** 1. ed. [S. l.]: McGraw Hill, 1997. 414p. Disponível em: <https://www.cs.cmu.edu/~tom/files/MachineLearningTomMitchell.pdf>. Acesso em: 01 abr. 2022.

PEARSON, K. LIII. On lines and planes of closest fit to systems of points in space. **The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science**, [S. l.], v. 2, n. 11, p. 559-572, nov. 1901. Disponível em: <https://doi.org/10.1080/14786440109462720>. Acesso em: 01 jul. 2023.

PÉREZ-BORRERO, I. *et al*. A fast and accurate deep learning method for strawberry instance segmentation. **Computers and Electronics in Agriculture**, v. 178, p. 105736, 2020. Disponível em: < https://doi.org/10.1016/j.compag.2020.105736>. Acesso em: 01 jul. 2023.

PSIROUKIS, V. Assessment of Different Object Detectors for the Maturity Level Classification of Broccoli Crops Using UAV Imagery. **Remote Sensing**, [S. l.], v. 14, n. 3, p. 731, 4 fev. 2022. Disponível em: <https://doi.org/10.3390/rs14030731>. Acesso em: 01 abr. 2022.

PYTHON. **Python 3.0 release**. Disponível em: <https://www.python.org/download/releases/3.0/>. Acesso em: 22 jan. 2024.

QUINATO, É. E.; DEGÁSPARI, C. H.; VILELA, R. M. Aspectos nutricionais e funcionais do morango. **Visão Acadêmica**, [S. l.], v. 8, n. 1, p. 11-17, 2007. Disponível em: <https://doi.org/10.5380/acd.v8i1.11660>. Acesso em: 31 dez. 2022.

RADOSAVOVIC, I. *et al*. Designing Network Design Spaces. **2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**, [S. l.], p. 10425-10433, jun. 2020. Disponível em: <https://doi.org/10.1109/CVPR42600.2020.01044>. Acesso em: 22 jan. 2024.

RAHMAN, M. *et al.* Maturity stages affect the postharvest quality and shelf-life of fruits of strawberry genotypes growing in subtropical regions. **Journal of the Saudi Society of Agricultural Sciences**, [S. l.], v. 15, n. 1, p. 28-37, 2016. Disponível em: <https://doi.org/10.1016/j.jssas.2014.05.002>. Acesso em: 31 dez. 2022.

ROBNIK-SIKONJA, M.; KONONENKO, I. Theoretical and empirical analysis of ReliefF and RReliefF. **Machine learning**, [S. l.], v. 53, n. 1/2, p. 23-69, 2003. Disponível em: <https://doi.org/10.1023/A:1025667309714>. Acesso em: 31 dez. 2022.

RODRIGUES, L. F. **Comparação entre Redes Neurais Convolucionais e técnicas de pré-processamento para classificar células HEp-2 em imagens de imunofluorescência.** 2018. 65 f. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Viçosa, Viçosa. 2018. Disponível em: <http://www.locus.ufv.br/handle/123456789/25441>. Acesso em: 01 abr. 2022.

ROKACH, L.; MAIMON, O. Decision Trees. **Data Mining and Knowledge Discovery Handbook**, [S. l.], p. 165-192, 2005. Disponível em: <https://doi.org/10.1007/0-387-25465-X_9>. Acesso em: 22 jan. 2024.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological Review**, [S. l.], v. 65, n. 6, p. 386-408, 1958. Disponível em: <https://doi.org/10.1037%2Fh0042519>. Acesso em: 22 jan. 2024.

RUSSAKOVSKY, O. *et al.* ImageNet large scale visual recognition challenge. **International Journal of Computer Vision**, [S. l.], v. 115, n. 3, p. 211–252, 2015. Disponível em: <https://doi.org/10.1007/s11263-015-0816-y>. Acesso em: 31 dez. 2022.

RUSSELL, S.; NORVIG, P. **Artificial intelligence**: a modern approach. 3. ed. Englewood Cliffs: Prentice Hall, 2009. 1132p. Disponível em: <https://people.engr.tamu.edu/guni/csce421/files/AI_Russell_Norvig.pdf>. Acesso em: 31 dez. 2022.

SANDLER, M. *et al*. MobileNetV2: inverted residuals and linear bottlenecks. **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**, [S. l.], p. 4510-4520, jun. 2018. Disponível em: <https://doi.org/10.1109/CVPR.2018.00474>. Acesso em: 22 jan. 2024.

SCHERER, D.; MÜLLER, A.; BEHNKE, S. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. **Artificial Neural Networks – Icann 2010**, [S. l.], p. 92-101, 2010. Disponível em: <https://amueller.github.io/papers/icann2010_maxpool.pdf>. Acesso em: 17 jan. 2024.

SCIKIT-LEARN, S. **Scikit-learn**: machine learning in python. Disponível em: <https://scikit-learn.org/stable/>. Acesso em: 22 jan. 2024.

SHETTY, D. *et al*. Diving Deep into Deep Learning: history, evolution, types and applications. **International Journal of Innovative Technology and Exploring Engineering**, [S. l.], v. 9, n. 3, p. 2835-2846, 30 jan. 2020. Disponível em: <https://doi.org/10.35940/ijitee.a4865.019320>. Acesso em: 22 jan. 2024.

SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: International Conference on Learning Representations, 2015. **Anais...** 2015. p. 1-14. Disponível em: <https://www.robots.ox.ac.uk/~vgg/research/very_deep/>. Acesso em: 31 dez. 2022.

SÜDHOF, T. C. The cell biology of synapse formation. **Journal of Cell Biology**, [S. l.], v. 220, n. 7, p. 1-18, 4 jun. 2021. Disponível em: <https://doi.org/10.1083/jcb.202103052>. Acesso em: 15 jan. 2024.

SUHARJITO; ELWIREHARDJA, G. N.; PRAYOGA, J. S. Oil palm fresh fruit bunch ripeness classification on mobile devices using deep learning approaches. **Computers and Electronics in Agriculture**, [S. l.], v. 188, p. 106359, 2021. Disponível em: <https://doi.org/10.1016/j.compag.2021.106359>. Acesso em: 31 dez. 2022.

SZEGEDY, C. *et al.* Rethinking the inception architecture for computer vision. **2016 IEEE Conference on Computer Vision and Pattern Recognition**, [S. l.], p. 2818-2826, 2016. Disponível em: <https://doi.org/10.1109/CVPR.2016.308>. Acesso em: 31 dez. 2022.

SZEGEDY, C. *et al*. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. **Proceedings of the AAAI Conference on Artificial Intelligence**, [S. l.], v. 31, n. 1, p. 4278-4284, 12 fev. 2017. Disponível em: <https://doi.org/10.1609/aaai.v31i1.11231>. Acesso em: 22 jan. 2024.

TAN, M.; LE, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In: CHAUDHURI, Kamalika; SALAKHUTDINOV, Ruslan (Ed.). **Proceedings of the 36th International Conference on Machine Learning**, 09-15 jun. 2019. PMLR, 2019. v. 97, p. 6105-6114. Disponível em: <https://proceedings.mlr.press/v97/tan19a/tan19a.pdf>. Acesso em: 22 jan. 2024.

TAN, M.; LE, Q. EfficientNetV2: Smaller Models and Faster Training. In: MEILA, Marina; ZHANG, Tong (Ed.). **Proceedings of the 38th International Conference on Machine Learning**, 18-24 jul. 2021. PMLR, 2021. v. 139, p. 10096-10106. Disponível em: <https://proceedings.mlr.press/v139/tan21a/tan21a.pdf>. Acesso em: 22 jan. 2024.

TEIXEIRA, J. F. **O que é inteligência artificial**. [S. l.]: Repositório Institucional da UFSC, 2009. 37p. Disponível em: <https://repositorio.ufsc.br/handle/praxis/395>. Acesso em: 01 abr. 2022.

TEMBUSAI, Z. R.; MAWENGKANG, H.; ZARLIS, M. K-nearest neighbor with k-fold cross validation and analytic hierarchy process on data classification. **International Journal of Advances in Data and Information Systems**, [S. l.], v. 2, n. 1, p. 1-8, 2021. Disponível em: <https://doi.org/10.25008/ijadis.v2i1.1204>. Acesso em: 31 dez. 2022.

VAPNIK, V.; LERNER, A. Pattern Recognition Using Generalized Portrait Method. **Automation and Remote Control**, [S. l.], v. 24, p. 774-780, 1963.

VIEIRA, D. F. A. **Catálogo brasileiro de hortaliças**: saiba como plantar e aproveitar 50 das espécies mais comercializadas no país. Brasília: Embrapa Hortaliças, 2010. 59p. Disponível em: <https://www.embrapa.br/busca-de-publicacoes/-/publicacao/887213/catalogo-brasileiro-

de-hortalicas-saiba-como-plantar-e-aproveitar-50-das-especies-mais-comercializadas-no-pais>. Acesso em: 01 abr. 2022.

WAN, P. *et al.* A methodology for fresh tomato maturity detection using computer vision. **Computers and electronics in agriculture**, [S. l.] v. 146, p. 43-50, 2018. Disponível em: <https://doi.org/10.1016/j.compag.2018.01.011>. Acesso em: 01 abr. 2022.

ZHANG, Y. *et al.* Deep indicator for fine-grained classification of banana's ripening stages. **EURASIP Journal on Image and Video Processing**, [S. l.], v. 2018, n. 1, p. 1-10, 2018. Disponível em: <https://doi.org/10.1186/s13640-018-0284-8>. Acesso em: 31 dez. 2022.

ZOPH, B. et al. Learning Transferable Architectures for Scalable Image Recognition. **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**, [S. l.], p. 8697-8710, jun. 2018. Disponível em: <https://doi.org/10.1109/CVPR.2018.00907>. Acesso em: 22 jan. 2024.