Uso de redes convolucionais para extração de características na busca de padrões de cores e texturas em folhas de oliveira

Gustavo Silva Quieregato

Dissertação apresentada à Escola Superior de Tecnologia e Gestão para obtenção do grau de mestre em Informática no âmbito da dupla diplomação com a Universidade Tecnológica Federal do Paraná

Orientadores:

Prof. Dr. Rui Lopes Lopes

Prof. Dra. Arlete Teresinha Beuren

Bragança

Março 2024

Dedicatória

Dedico este projeto às pessoas fundamentais em minha vida: meus pais, Walter, Janaina e ao meu padrastro Paulo, que sempre me apoiaram em cada sonho, me orientando e dando todo o suporte para conquistar cada um dos sonhos, aos meus avós, mas que também sempre me mostrou que vale a pena lutar pelos objetivos, apesar das dificuldades. Aos meus amigos, que me incentivaram durante toda a jornada. Um grande agradecimento a professora Arlete, cuja orientação foi fundamental para este trabalho e também para a vida profissional.

Agradecimentos

Gostaria de expressar minha mais profunda gratidão e reconhecimento a todas as pessoas que me acompanharam ao longo da minha jornada acadêmica. Em primeiro lugar, quero expressar minha gratidão a Deus por guiar meus passos até este momento, sendo fonte de luz e sabedoria em cada etapa do caminho.

Um agradecimento especial aos meus pais, Walter, Janaina, Paulo, que estiveram ao meu lado desde o início da minha trajetória acadêmica e profissional. Agradeço pelo amor incondicional, pelos ensinamentos transmitidos ao longo dos anos e por me inspirarem a acreditar que, independentemente dos desafios, é possível realizar sonhos. Vocês são exemplos de vida para mim. Também quero agradecer a minha futura esposa, Stefany, por todo apoio e principalmente por acreditar que seria possível. Desejo também expressar minha gratidão às instituições que tornaram tudo isso possível: a Universidade Tecnológica Federal do Paraná e o Instituto Politécnico de Bragança. Essas oportunidades representaram marcos significativos em minha vida, permitindo-me vivenciar experiências únicas, como a descoberta de novas culturas, idiomas, pessoas e conhecimentos. Também gostaria de expressar minha gratidão à minha namorada, Stefany Viveiros Barboza, fonte de amor e incentivo para alcançar cada um dos meus sonhos, sendo meu porto seguro em todos os momentos. Aos meus orientadores, Rui Pedro Lopes e Arlete Teresinha Beuren, dirijo minha mais sincera gratidão. Suas orientações foram fundamentais para superar barreiras do conhecimento, possibilitando meu crescimento pessoal e profissional. Agradeço também pela confiança depositada em mim e por compartilharem seu vasto conhecimento.

Por fim, gostaria de agradecer ao Arthur, Luiz e o Jean, que ao longo dessa jornada

no mestrado, estiveram me incentivando a continuar e me ensinaram muito.

A todos que de alguma forma contribuíram para minha jornada acadêmica, meu muito obrigado. Essa conquista é também de vocês.

Resumo

Atualmente, é possível visualizar uma abundância de pesquisadores em busca de estratégias aprimoradas para identificação de cores e texturas em imagens, seja por métodos analíticos, espectrais ou estruturais. Nesse contexto, a presente dissertação propõe uma análise teórica e aprofundada dos modelos de redes convolucionais, com foco nas camadas mais profundas da rede neural residual ResNet50. O principal objetivo é identificar as diferentes sensibilidades existentes sobre cada um dos blocos residuais presentes no modelo de rede neural ResNet50, utilizando diferentes técnicas de pré-processamento sobre o conjunto de imagens de Folhas de Oliveiras (saudáveis e doentes).

Para atingir esse propósito, a pesquisa não apenas irá apresentar um estudo teórico das metodologias de Inteligência Artificial, mas também apresentará uma análise aplicando transformações sobre as diversas tonalidades presentes nas imagens. Neste contexto, será utilizado essa metodologia visando aprimorar o entendimento das características visuais das folhas de oliveira, proporcionando uma análise mais completa sobre a eficácia das camadas de convolução no processo de classificação. Além disso, a dissertação visa contribuir para o avanço da pesquisa ao explorar diferentes estratégias na análise de texturas, aprimorando a capacidade de diferenciação entre folhas saudáveis e doentes, com potenciais aplicações em monitoramento agrícola e diagnóstico precoce de doenças.

Palavras-chave: Redes Convolucionais, ResNet50, Folhas de oliveira, Textura e Classificação

Abstract

Currently, it is possible to see a large number of researchers looking for improved strategies for identifying colors and textures in images, whether through analytical, spectral or structural methods. In this context, this dissertation proposes an in-depth theoretical analysis of convolutional network models, focusing on the deepest layers of the ResNet50 residual neural network. The main objective is to identify, through a set of images, the textures present and categorize the layers according to their performance in classifying olive leaves (healthy and diseased).

To achieve this purpose, the research will not only present a theoretical study of Artificial Intelligence methodologies, but will also present an analysis applying transformations to the different tones present in the images. In this context, this methodology will be used with the aim of improving the understanding of the visual characteristics of olive leaves, providing a more complete analysis of the effectiveness of convolution layers in the classification process. Furthermore, the dissertation seeks to contribute to the advancement of research by exploring different strategies in texture analysis, improving the ability to differentiate between healthy and diseased leaves, with potential applications in agricultural monitoring and early diagnosis of diseases.

Keywords: Convolutional Networks, ResNet50, Olive Leaves, Texture and Classification

Conteúdo

Lista de Tabelas

Lista de Figuras

Acrónimos

AHE Adaptive Histogram Equalization. 27

CDF Cummulative Distribution Function. 27

CIE Comission Interlationale de Eclairage. 20

CLAHE Contrast Limited Adaptive Histogram Equalization. 27, 32, 59

CMYK Cyan, Magenta, Yellow, Black. xii, 25, 26

CNN Convolutional Neural Network. 1, 14, 17

CV Computer Vision. 19

CWI Fundação Centro de Matemática. 45

DL Deep Learning. 13

GHE Global Histogram Equalization. 27

GUI Graphical User Interface. 45

HE Histogram Equalization. 26

HSV Hue, Saturation and Value. 20, 21, 23, 25

IA Inteligência Artificial. 5, 6, 13, 19

KNN K-Nearest Neighbors. 28, 32, 49, 75

LHE Local Histogram Equalization. 27

ML Machine Learning. 6, 7

QI Quociente Intelectual. 5

RGB Red, Green and Blue. xii, 20, 22, 23, 25, 26

RNA Redes Neurais Artificiais. 8–10, 74

TL Transfer Learning. 18

XOR Ou exclusivo. 11

Capítulo 1

Introdução

O capítulo 1 descreve o enquadramento e contexto da realização desta dissertação, a sua motivação, objetivos e estrutura do documento.

1.1 Enquadramento

No cenário atual de ciência de dados e inteligência artificial, a aplicação de algoritmos de aprendizado de máquina para classificar dados é cada vez mais importante. A capacidade de classificar dados em diferentes classes com base nas suas características intrínsecas tem implicações de longo alcance numa variedade de campos, desde medicina e finanças até reconhecimento de padrões e processamento de linguagem natural.

Este trabalho enquadra-se neste contexto, visando explorar e comparar o desempenho de diferentes blocos de redes neurais convolucionais Convolutional Neural Network (CNN), especialmente na arquitetura ResNet50, diante diferentes desenvolvimentos de imagens. O objetivo é avaliar a sensibilidade desses blocos às mudanças nos dados de entrada, buscando compreender melhor como se comportam os modelos residuais. O objetivo desta tese é, assim, realizar análises detalhadas a modelos de redes convolucionais, com foco nas camadas mais profundas do modelo ResNet50, para perceber como os diferentes blocos respondem às variações da imagem e como essa sensibilidade pode afetar o desempenho geral da rede.

Adicionalmente, serão exploradas técnicas de pré-processamento de dados, como normalização, tratamento de valores faltantes e seleção de atributos, para melhorar o desempenho do algoritmo e evitar *overfitting* e *underfitting*. Esta abordagem visa garantir que os modelos possam generalizar para amostras eficazmente, independentemente das características específicas dos dados de entrada.

1.2 Objetivos

O objetivo desta dissertação é conduzir uma análise detalhada dos modelos de redes convolucionais, com foco nos diversos blocos de convolução, especialmente nas camadas do modelo de rede neural residual ResNet50. A pesquisa visa criar diferentes conjunto de imagens, utilizando como base um dataset já construído e disponível, aplicando diferentes técnicas de pré-processamento. Neste contexto, pretende-se coletar os blocos residuais do modelo da Resnet50, verificando a sensibilidade sobre cada uma das técnicas aplicadas.

Para a implementação do objetivo geral, os seguintes objetivos específicos são definidos:

- Estudar os blocos de convoluções existentes no modelo escolhido.
- Aplicar variações na cor, textura e brilho de forma a gerar novos datasets de imagens.
- Treinar cada bloco de convolução utilizando diferentes conjuntos de dados.
- Avaliar as métricas escolhidas após o treinamento.

1.3 Estrutura do Documento

A dissertação está estruturada da seguinte forma: o **capítulo 2** apresenta a fundamentação teórica sobre os problemas de identificação de cores e texturas, uma introdução sobre Inteligência Artificial, Aprendizado de Máquina, Redes Neurais e Deep Learning, bem como serão apresentados os trabalhos relacionados. No **capítulo 3**, serão apresentados detalhamentos sobre os processos de extração das camadas necessárias para a realização

dos testes, introduzidas as diferentes técnicas para a construção do conjunto de dados, e a implementação e coleta das métricas de treinamento. O **Capítulo 4**, será apresentada uma análise dos resultados encontrados, separando-os conforme os conjuntos de dados testados; finalmente, o último capítulo apresentará as conclusões e direções de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

O capítulo 2 descreve e exemplifica os conceitos necessários para o entendimento completo deste trabalho de dissertação. Será realizado uma breve introdução sobre a importância do estudo prático de cores e texturas em diferentes situações, utilizando técnicas de pré-processamento, apresentando outros termos como Inteligência Artificial, Modelos de *Machine Learning*, *Deep Learning*

2.1 Inteligência Artificial

O termo Inteligência Artificial (IA) foi introduzido por John McCarthy em 1955, sendo um grande marco para o início de uma área da computação dedicada a replicar, de maneira sintética, o aprendizado humano, assim como as ações e interpretações em diversos ambientes. Desde então, a IA tem evoluído consideravelmente, abrangendo uma variedade de técnicas e abordagens para simular processos cognitivos e comportamentos inteligentes.

Existem diversos elementos que devem ser considerados para que o objetivo das IA consigam ser alcançado, dentre eles o que mais se destaca é a compreensão profunda da inteligência. Travessos (2010) propõe uma visão mais tradicional, onde a inteligência é considerada a capacidade inata do indivíduo, possibilitando a realização de testes como o Quociente Intelectual (QI) [1]. No entanto, existe uma perspectiva Piagetiana, que enfatiza a construção do conhecimento por meio das ações do sujeito. Flavel (1996) argumenta

que a inteligência não está estritamente atrelada ao processo de herança biológica [2].

Além disso, Russell (2016), destaca que a IA engloba diversas abordagens, desde sistemas baseados em regras e aprendizado de máquina até redes neurais e algoritmos evolutivos. O aprendizado de máquina, por exemplo, é uma subárea crucial da IA, onde os algoritmos podem aprender padrões a partir de dados e aprimorar seu desempenho ao longo do tempo [3].

A IA tem aplicações práticas em diversos setores, como saúde, finanças, automação industrial e tecnologia da informação [4]. A busca contínua por algoritmos mais avançados e aprimoramento das técnicas de IA levam a avanços significativos, mas também levantam questões éticas, sociais e legais, exigindo uma reflexão contínua sobre o impacto da IA na sociedade [5], [6].

Portanto, a IA, desde sua origem em 1955, tem-se transformado em uma área dinâmica e interdisciplinar, moldando não apenas o campo da computação, mas também influenciando profundamente como interagimos com a tecnologia e enfrentamos desafios complexos em diversas esferas da vida cotidiana [7].

2.1.1 Aprendizagem Máquina

O *Machine Learning (ML)*, ou aprendizado de máquina em português, é uma área da IA, que por meio da criação de algoritmos, permite ao computador identificar padrões em quantidades massivas de informações, realizar análises preditivas, entre outras tarefas de reconhecimento.[8]–[11].

A integração destas capacidades computacionais avançadas é importante em muitos campos, permitindo avanços significativos em áreas como medicina, robótica e construção. Na área da saúde, por exemplo, o aprendizado de máquina possibilita realizar análise de dados clínicos, no diagnóstico médico e na descoberta de padrões em grandes conjuntos de dados [3]. Um estudo de Rajkomar (2018), demonstra uma aplicação utilizando métodos de ML para prever resultados clínicos, fornecendo informações importantes para os profissionais de saúde [12].

Na robótica, o aprendizado de máquina é frequentemente usado para melhorar a autonomia e permitir aos robôs se adaptarem a ambientes dinâmicos. Um estudo realizado por Kober (2013) [13] explora técnicas de aprendizagem por reforço para melhorar o controle do robô em tarefas complexas e mutáveis. Na construção, o ML é usado para otimizar o planejamento e a execução do projeto, analisando dados históricos para prever custos, identificar riscos e melhorar a eficiência geral do processo.

Outro estudo realizado por Mbachu (2020), demonstra como as técnicas de ML podem ser aplicadas ao gerenciamento eficaz de projetos de construção [14]. A utilização deste modelo de aprendizagem em meio a vários campos de estudo destaca não só a diversidade dos dados, mas também um grande potencial em meio aos avanços importantes que beneficiam a sociedade [4].

Abaixo estão listadas alguns dos modelos de aprendizado existentes (Figura 2.1).

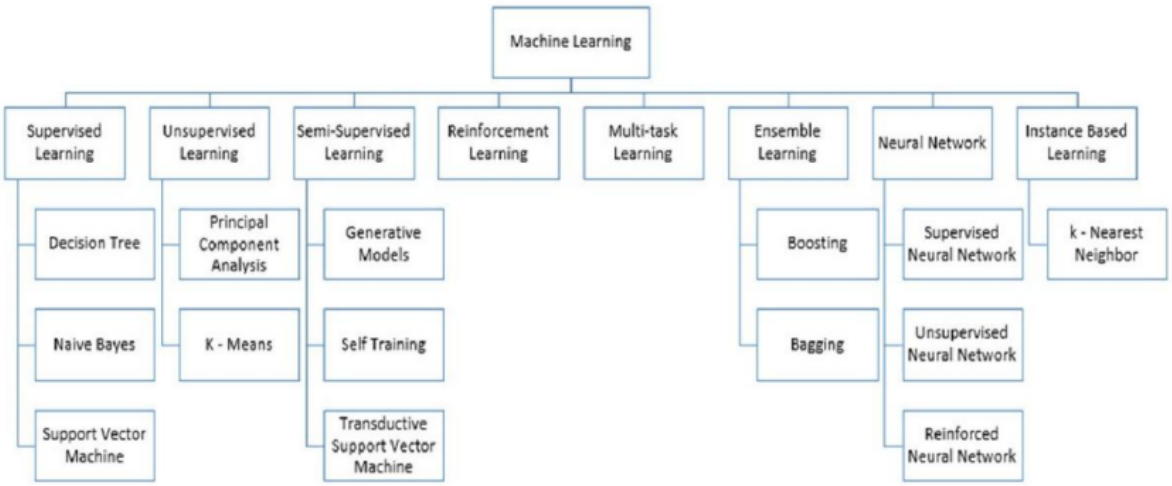


Figura 2.1: Machine Learning [9]

- **Aprendizado Supervisionado**: Este modelo de aprendizado utilizado uma estratégia de treinamento com base nos dados devidamente conhecidos e rotulados, ou seja, o objetivo é conseguir encontrar uma hipótese válida para o conjunto de dados, permitindo que o modelo consiga classificar corretamente os exemplos de teste [8].
- **Aprendizado Não Supervisionado**: Os modelos que utilizam o aprendizado

não supervisionado, ou seja, não possuem rótulos nos exemplos de treinamento, deixando com que o modelo realize a identificação com base em sua representação, e características [8], [15].

- **Aprendizado por Reforço**: Diferentemente das formas de aprendizado introduzidas anteriormente, utiliza uma abordagem mais interativa, sendo necessário obter o auxílio de um agente externo. O aprendizado por reforço, tende a utilizar ganhos e punições para conseguir chegar ao objetivo determinado, desta forma o modelo irá conseguir estabelecer determinadas políticas que tornam a quantidade de ganhos maiores do que as punições [16].

2.1.2 Redes Neurais

Redes Neurais Artificiais (RNA) são modelos computacionais que se inspiram no sistema humano de processamento de informações, utilizando neurônios artificiais para formar uma estrutura organizada entre o sistema de entrada, processamento interno e saída. Inicialmente desenvolvido por McCulloch e Pitts [17], [18], esse paradigma de inteligência artificial tem sido amplamente explorado e aplicado em diversas áreas do conhecimento, a Figura 2.2 apresenta um exemplo simples desta arquitetura de rede neural artificial.

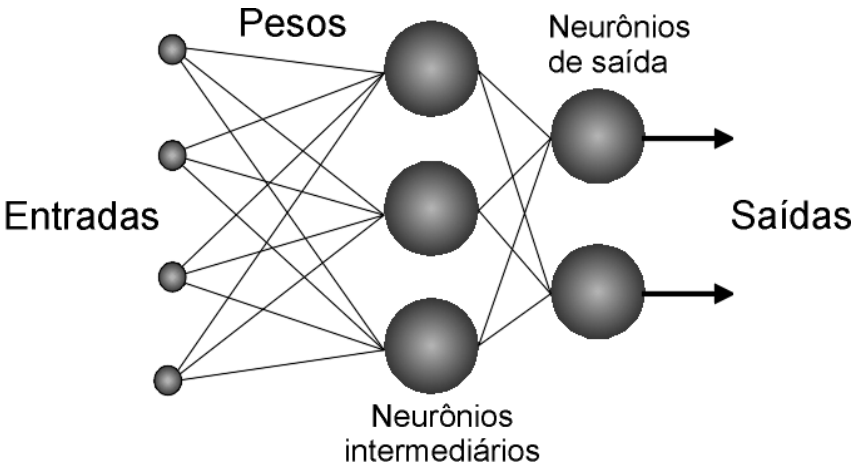


Figura 2.2: Exemplo de rede neural artificial [19]

GoodFellow (2016) explica sobre os modelos de RNA que oferecem determinada flexibilidade, ao serem usados em diversos campos, incluindo medicina, finanças e reconhecimento de padrões. A capacidade de estudar o comportamento e entender como identificar padrões nos dados de entrada é essencial para o uso eficaz das RNAs em problemas reais [10].

A estrutura de uma RNA pode variar dependendo do modelo escolhido. Uma característica única é que existe uma camada intermediária onde ocorre o processo de aprendizagem. Nesta camada, muitas conexões ocorrem entre os neurônios conforme os princípios de aprendizagem humana. Os neurônios são ativados ou desativados dependendo das sinapses recebidas, ajustando seus pesos para otimizar a capacidade da RNA de realizar uma determinada tarefa [20].

O estudo e a compreensão aprofundada das Redes neurais artificiais proporcionam parâmetros de análise valiosos sobre o funcionamento do cérebro artificial e impulsionam avanços contínuos na área de inteligência artificial [4]. A aplicação prática das RNAs em diversas disciplinas destaca a importância crescente desses modelos na resolução de problemas complexos e na melhoria de sistemas autônomos e adaptativos. Ao explorar suas nuances e adaptá-las às demandas específicas, os pesquisadores podem aproveitar ao máximo o potencial das redes neurais artificiais [4], [10].

Estruturas Biológicas

O sistema nervoso humano é constituído por uma complexa rede de neurônios, desempenhando um papel crucial na transmissão de sinais elétricos, também conhecidos como pulsos, entre as células nervosas (Figura 2.3). Cada neurônio apresenta uma estrutura complexa, sendo melhor compreendida se dividida em partes, conforme destacado por Luo et al. (2021) [21].

- Dendritos [22]: possuem ramificações numerosas, que se estendem durante todo o corpo celular do neurônio, estes são responsáveis pela recepção de cada estímulo recebido.

- Axônio [22]: São responsáveis pela transmissão dos estímulos, ao contrário dos dendritos, existe apenas 1, para cada neurônio.
- Sinapse [22]: É descrito sendo a junção funcional entre o terminal axonal de um neurônio (pré-sináptico) e os dendritos (pós-sinápticos).
- Propagação do Impulso Nervoso [22]: É o potencial de ação que viaja ao longo de todo o axônio, até os terminais axonais, ao final irá realizar o processo de liberação dos neurotransmissores para cada sinapse estimulada.

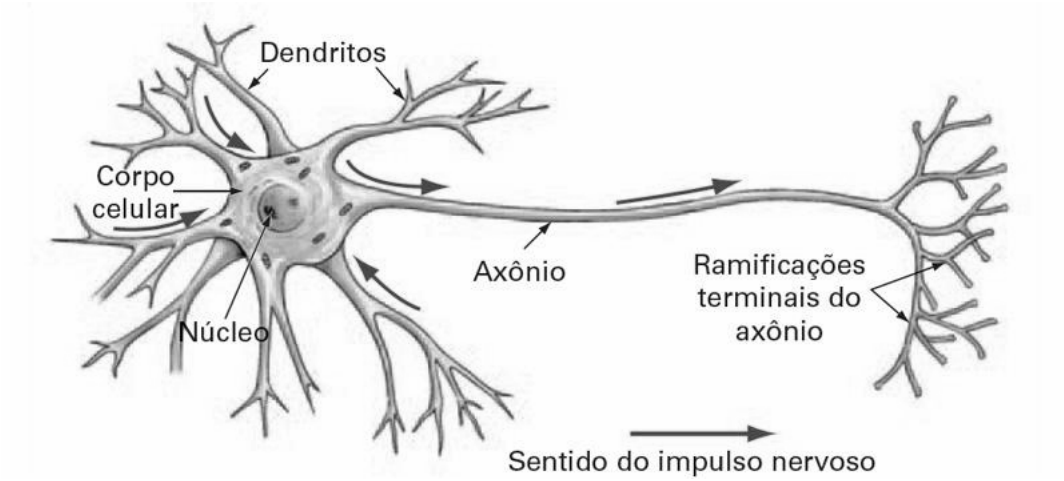


Figura 2.3: Neurônio biológico [23]

Dessa maneira, torna-se evidente a presença de uma organização estrutural análoga nas RNA (Figure 2.4). Nesse contexto, os axônios desempenham o papel de representar as conexões entre os nós componentes da rede, os dendritos podem ser associados às entradas recebidas pelos neurônios, enquanto o corpo celular encontra sua representação na função de ativação.

Redes Perceptron e Multilayer Perceptron

Os primeiros modelos de redes neurais, como o perceptron de Frank Rosenblatt em 1958, desenvolveram uma abordagem básica de entrada-saída focada na resolução de problemas lineares. Experimentos pioneiros com **MARK I PERCEPTRON** marcaram a época,

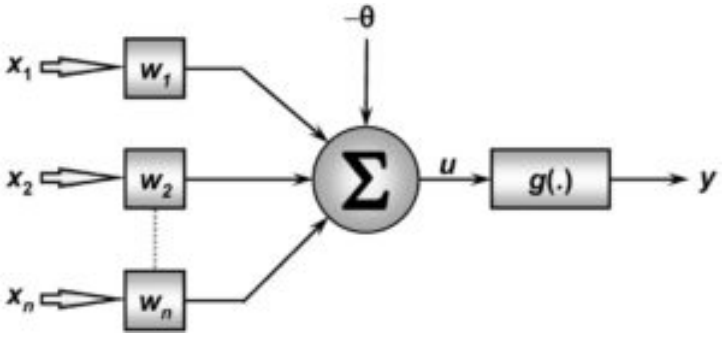


Figura 2.4: Representação artificial de um neurônio [24]

levantando grandes questões que poderiam revolucionar a interação entre humano e computadores. Ele opera sobre um algoritmo de aprendizagem, sendo projetado para calcular pesos internos para resolver um problema específico [17], [25].

- **Desafios na interação humano-computador**: Surgimento de questões sobre a possibilidade de explorar mais profundamente as capacidades dos modelos [26].
- **Limitações na resolução de problemas lineares**: Os primeiros modelos de redes neurais foram desenvolvidos para resolver problemas lineares, limitando-os em problemas mais complexos [17], [26].
- **Desenvolvimento de algoritmos de aprendizado**: Desenvolvimento de novos algoritmos de aprendizagem para conseguirem resolver problemas variados de forma mais eficiente [17], [26].

Se aplicado a problemas como o Ou exclusivo (XOR), este problema destaca-se ainda mais. Sendo descoberto mais tarde que o mesmo poderia ser superado adicionando uma camada oculta, chamada camada intermediária ou camada oculta. Desde então, uma série de experimentos e estudos foram conduzidos na tentativa de desenvolver modelos avançados que possam resolver problemas mais complexos, resultando diretamente nas arquiteturas mais complexas. Este período de pesquisa e inovação marcou um grande progresso na compreensão e evolução das redes neurais.

No contexto das camadas cruzadas, a descoberta de algoritmos mais complexos e o entendimento mais profundo da sua complexidade inerente levaram ao desenvolvimento

de redes neurais que podem lidar com tarefas cada vez mais complexas. Esta evolução é evidenciada pelo surgimento de arquiteturas modernas, como as redes neurais profundas, que mostram excelentes resultados mediante ao desempenho em aplicações que vão desde o reconhecimento de padrões até o processamento de linguagem natural. Melhorias e inovações contínuas neste campo.

Função de Ativação

As funções de ativação desempenham um papel crucial no processo de aprendizado de modelos em redes neurais, tendo dois estados fundamentais: o neurônio ativo e o desativado. Este processo dinâmico varia conforme as entradas das camadas iniciais da rede, interferindo diretamente na capacidade de aprendizado e adaptação do modelo ao longo do tempo.

A complexidade e diversidade dos padrões existentes nos dados são aprendidos pelos neurônios por meio das funções matemáticas específicas. Neste contexto, deve-se ter o cuidado durante a escolha destas funções, bem como a escolha do modelo [27].

A capacidade de ajuste oferecida pelas funções de ativação desempenha um papel crucial na eficácia geral da rede neural em aprender e representar informações de maneira significativa [28]. Este processo de adaptação é fundamental para conseguir solucionar os desafios presente em diversas aplicações, desde reconhecimento de padrões até processamento de linguagem natural [28].

Existem diversas categorias de funções que podem ser utilizadas em redes neurais, como: funções lineares, de degrau e sigmoidais. Conforme Rauber (2005), a função Sigmoid, também conhecida como função logística, emprega logaritmos naturais para converter qualquer valor de entrada em um intervalo restrito entre 0 e 1 [28]. O comportamento característico desta função pode ser visualizado na Figura 2.5. A delimitação desses intervalos possibilita a redução da dimensionalidade do espaço de trabalho, proporcionando uma visualização mais clara sobre o comportamento dos dados [28].

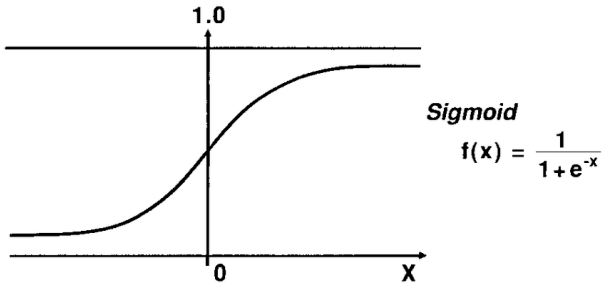


Figura 2.5: Função de ativação Sigmoid [28]

2.1.3 Aprendizagem Profunda

O Deep Learning (DL), um subcampo do aprendizado de máquina, visa aprender modelos de IA baseados no comportamento do cérebro humano usando redes neurais artificiais. Este modelo pode usar camadas ocultas [10] para processar informações de forma mais sofisticada.

A estrutura do aprendizado profundo pode ser comparada a um grafo, podendo ser utilizado em diferentes níveis e conjuntos de modelos [11]. Cada camada de uma rede neural profunda atua como uma transformação de dados, aprendendo padrões cada vez mais complexos à medida que é realizado o processo pelas camadas ocultas. Essa hierarquia de representação é essencial para o aprendizado profundo conseguir aprender recursos complexos de dados de alto nível.

A utilização dessas estruturas em Aprendizado Profundo é motivada pela habilidade desses modelos de criar abstrações das informações obtidas, gerando representações que encapsulam características essenciais dos dados [11]. Essa capacidade de abstração é crucial para a generalização eficaz do modelo em tarefas diversas, permitindo que ele reconheça padrões complexos e se adapte a novos conjuntos de dados.

No trabalho de Schmidhuber [11] é destacada a importância de estruturas hierárquicas e abstratas para a eficácia do Aprendizado Profundo. Ao proporcionar uma visão detalhada sobre as representações e transformações existentes, o Aprendizado Profundo é destacado como uma ferramenta poderosa para compreender e lidar com a complexidade presente nas informações existentes.

Redes Neurais Convolucionais

As CNN surgiram como arquiteturas de aprendizagem profunda, sendo eficientes em tarefas por meio de visão computacional e reconhecimento de padrões em imagens [29]. Essas redes incluem camadas convolucionais e de pooling, que permitem extrair recursos hierárquicos e também permitem reduzir a dimensionalidade dos dados. Um trabalho importante que destaca a importância das CNN na visão computacional é o de Krizhevsky, Sutskever e Hinton [29], sendo demonstrado o funcionamento de uma CNN chamada de AlexNet. Krizhevsky (2012), menciona que a presença de camadas totalmente conectadas em uma CNN é importante, permitindo que as informações sejam obtidas durante o processo de aprendizagem. Estas camadas são responsáveis por combinar características aprendidas nas camadas anteriores, facilitando assim a representação de informações complexas e não lineares. A pesquisa de Simonyan e Zisserman [30], apresenta uma arquitetura chamada de VGGNet, do qual permite que seja visualizado o impacto positivo das camadas ocultas em meio as CNN.

O uso das CNN permite que sejam aplicadas em diversas áreas, desde reconhecimento de objetos até segmentação de imagens e reconhecimento facial [29], [31]. Trabalhos recentes, como o de He, Zhang, Ren e Sun (2016), apresentam a arquitetura ResNet, introduzindo os blocos residuais após conseguir solucionar alguns dos problemas existentes, como desaparecimento de gradientes, proporcionando melhor desempenho em tarefas de classificação de imagens [32]. Outra característica das CNN é a capacidade de realizar aprendizagem por transferência, permitindo a utilização de pesos pré-treinados por modelos em tarefas semelhantes. Este processo se mostra útil em casos onde o conjunto de dados de treinamento é limitado [31]. Yosinski (2014), fornece uma análise aprofundada sobre a possibilidade de utilizar tais recursos nas CNN [33], considerando a eficácia desse processo.

Métricas de avaliação

As métricas de avaliação desempenham um papel importante durante a análise de desempenho dos modelos durante o processo de treinamento, sendo aplicadas tanto em problemas de classificação quanto de regressão [25], [34]. Estas métricas utilizam de ferramentas essenciais para avaliar a precisão, confiabilidade e generalização dos modelos gerados.

A escolha da métrica irá depender da natureza do problema em questão, sendo comum a utilização de mais de uma métrica para conseguir avaliar o modelo [34]. Isso permite uma compreensão aprofundada dos resultados, podendo obter vários pontos de observação do modelo, resultando em uma análise mais confiável e precisa [34].

Uma das métricas que normalmente é utilizada, é chamada de matriz de confusão, que fornece uma visão detalhada das relações entre as classes no conjunto de dados. Esta matriz é composta por quatro valores principais: Verdadeiro Positivo (VP), Verdadeiro Negativo (VN), Falso Positivo (FP) e Falso Negativo (FN) [10], [34]. A Figura 2.6 mostra visualmente esses elementos, podendo então obter uma visualização clara do desempenho do modelo em diferentes cenários.

Além dessas métricas tradicionais, existem métricas específicas para problemas particulares, como sensibilidade, especificidade e precisão, que podem oferecer análises mais detalhadas dependendo do contexto em que é aplicado [25].

- **Verdadeiro Positivo (VP)**: Representam a quantidade de instâncias da classe positiva que foram classificadas corretamente.
- **Verdadeiro Negativo (VN)**: Representam a quantidade de instâncias da classe negativa que foram identificadas como negativas.
- **Falso Negativo (FN)**: Representa a quantidade de instâncias da classe positiva que foram classificadas erradamente pelo modelo como negativas.
- **Falso Positivo (FP)**: Representa a quantidade de instâncias da classe negativa que foram classificadas erradamente pelo modelo como positivas.

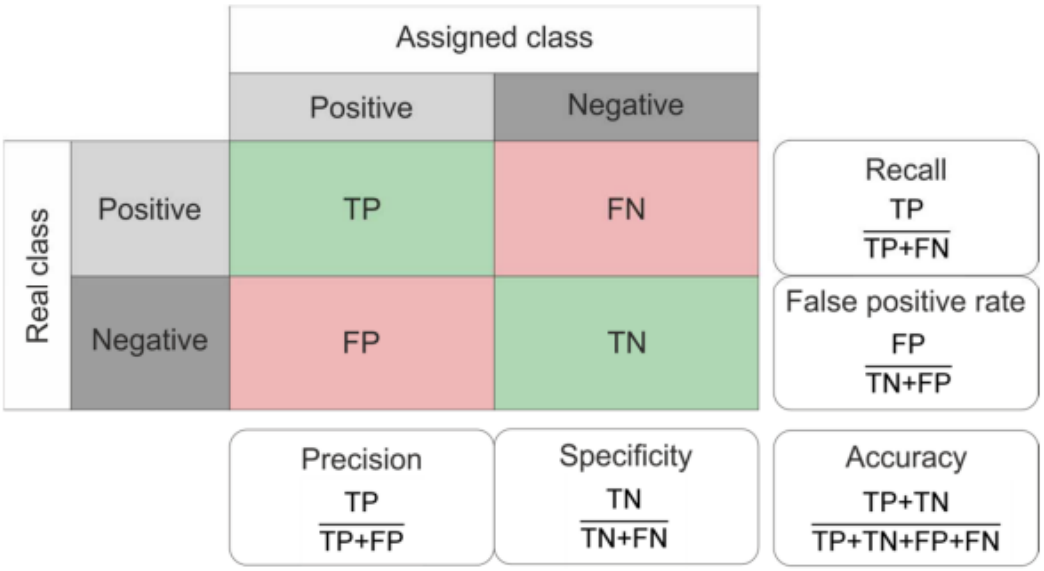


Figura 2.6: Construção da matriz de confusão[35]

Neste contexto, existem outras métricas que auxiliam durante esse processo de análise, das quais são descritas logo abaixo:

- Acurácia - É utilizada como uma métrica central para a avaliação dos classificadores, desta forma, é possível medir a eficiência de uma inteligência artificial. No entanto, segundo Junior (2022), Rothe e Sandra (2016) [34], [35], a acurácia não deve ser utilizada unicamente para medir tais princípios de desempenho. Este valor pode ser calculado mediante ao total de observações verdadeiras obtidas (negativas e positivas) dividido pela soma dos elementos existentes [34].

- Especificidade - Avalia a taxa de verdadeiro negativo existentes em um classificador, sendo calculada a partir dos (VN) em razão da soma dos verdadeiros negativos como também dos falso positivos (VN+FP)[34], conforme visto na Figura 2.6.

- Precisão - Utilizada para medir o impacto dos valores de falso positivo (FP) sobre os falso negativos (FN), assim, garantindo que tal erro seja o menor possível[34], como é possível visualizar na figura 2.6.

Arquitetura da CNN

Atualmente, existem diversas arquiteturas de CNNs que estão disponíveis para resolver diferentes categorias de problemas. Neste contexto, destaca-se a existência de redes neurais, com grande capacidade de aprender e de transferir informação entre diferentes níveis de processamento [29].

Dentre essas arquiteturas, a ResNet50 destaca-se dentre os modelos existentes, tendo em vista a sua capacidade de treinamento e estrutura de montagem. A arquitetura da ResNet foi construída para resolver problemas comuns em treinamento profundo, como perda de gradiente [32]. O modelo utilizando 50 camadas da ResNet possui uma arquitetura mais profunda e complexa, contribuindo para a eficiência do aprendizado para representações complexas de dados visuais. Ao resolver problemas específicos de visão computacional, a ResNet50 oferece vantagens significativas [32].

ResNet50

ResNet é uma abreviatura de Residual Neural Network e representa uma arquitetura de rede neural profunda na área do aprendizado profunda. Foi desenvolvido pela Microsoft Research e pertence à família de redes neurais residuais. É caracterizado por sua capacidade única de solucionar problemas associados ao desaparecimento de gradientes. Isso foi possível por meio da introdução de um bloco residual que consiste em uma combinação e uma função residual integrada na construção do modelo [32].

Com a apresentação da ResNet, iniciam-se reduções significativas nos erros de treinamento em redes neurais profundas. As ligações de enlace que permitem a transferência direta de informação de uma camada para outra, e a função residual que representa a diferença entre a entrada e a saída de um bloco, permitindo otimizar o treinamento destes modelos [10], [32].

A ResNet50, utiliza 50 camadas, sendo 48 camadas de convolução, uma camada de pooling máximo e uma camada de pooling médio [36]. Esta configuração é dividida em cinco blocos convolucionais diferentes, cada um com propriedades específicas para extrair

representações complexas de dados visuais. Na Figura 2.7 é possível visualizar os detalhes para cada estrutura.

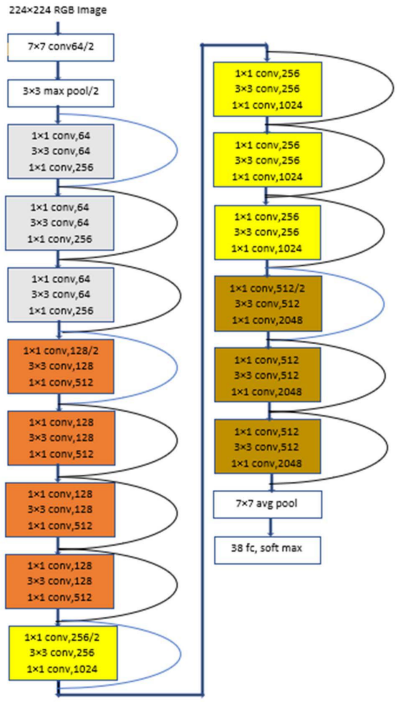


Figura 2.7: Arquitetura da ResNet50 [31]

2.1.4 Aprendizagem por Transferência

O *Transfer Learning (TL)*, também conhecido como Aprendizado por Transferência, é uma estratégia para o aprendizado de máquina, conforme Pan e Yang (2010) em seu estudo sobre o tema [37]. Esta abordagem utiliza do conhecimento adquirido em outros processos de treinamento, melhorando o desempenho das tarefas relacionadas. O uso desta estratégia pode ser utilizado em diversas áreas, desde a identificação de flores e objetos de varejo até a categorização de automóveis, dentre outros tipos categóricos [38].

A implementação do TL em modelos de aprendizado profundo, conforme abordado por Weiss, Khoshgoftaar e Wang em sua pesquisa sobre transfer learning [39], torna o processo de treinamento mais rápido e com maior eficiência de processamento. Sendo assim é possível visualizar no experimento realizado por Bengio, Courville e Vicent (2013) durante

uma análise sobre os algoritmos de representação, mostrando ferramentas essenciais para a otimização do desempenho dos modelos de aprendizado de máquina [40].

2.1.5 Visão por Computador

A Computer Vision (CV) é um campo da IA projetado para auxiliar os computadores a interpretar e compreender o mundo visual da mesma forma que os humanos [41]. Este campo de pesquisa inclui diversas técnicas e algoritmos para analisar e extrair informações significativas de imagens e vídeos [41]. Em Szeliski (2010) é destacado a intensa quantidade de pesquisadores, em busca de abordagens para alcançar essas capacidades computacionais [42]. Estas incluem técnicas avançadas em processamento de imagens, aprendizado de máquina, redes neurais convolucionais, etc.

Um dos principais objetivos da visão computacional é treinar os sistemas computacionais para executar tarefas específicas, como reconhecimento e classificação de objetos, reconhecimento de padrões, reconhecimento facial, detecção de movimento e segmentação de imagens. Neste contexto, pode ser aplicado a diversas áreas, como: saúde, indústria, segurança e entretenimento [41].

Podendo citar como exemplo a indústria, a integração de sistemas de visão computacional possui um grande impacto sobre a produtividade e na eficiência operacional [41]. Automatizando os processos de inspeção de qualidade, controle de produção e manutenção preditiva dos alimentos por sistemas inteligentes de visão computacional, reduzindo custos e aumentando os lucros [41].

A visão computacional também desempenha um papel fundamental nos avanços tecnológicos, como carros autônomos, realidade aumentada, medicina e segurança. À medida que a pesquisa continua avançando e a tecnologia se torna mais acessível [43].

2.2 Efeito da Cor na Visão por Computador

Os sistemas computacionais dependem de uma série de cálculos matemáticos para interpretar dados e criar imagens digitais. Esses processos dependem de como os humanos

percebem as imagens. Segundo estudos como [44], [45], a percepção das cores envolve uma combinação de três elementos, chamados tristímulos: R (vermelho), G (verde) e B (azul). Representadas pelas cores primárias ou componentes básicos que utilizam espaços de cores lineares e não lineares.

Existem vários sistemas de representação de cores atualmente em uso, incluindo RGB, Hue, Saturation and Value (HSV) e Comission Interlationale de Eclairage (CIE). Cada um desses sistemas oferece uma abordagem diferente para representar e processar cores em imagens digitais, com vantagens e aplicações específicas [46]. Pode-se utilizar como exemplo o modelo RGB utilizado em dispositivos de exibição, como monitores de computador e TV, enquanto o modelo HSV é utilizado em aplicativos de processamento de imagem, que ajudam a manipular cores com base em matiz, saturação e valor com mais facilidade [44].

Além destes sistemas de cores, existem muitas outras técnicas e modelos que exploram diferentes aspectos da percepção visual humana e das propriedades da luz [45]. De acordo com Gonzales (2008) é essencial entender a aplicação dos diferentes sistemas de cores, bem como suas diferentes aplicações em meio a visão computacional [43].

2.2.1 Espaços de cores

O sistema de cores HSV é usado para representar espaços de cores de uma forma mais intuitiva ao olho humano do que o sistema de cores RGB (Figura 2.8). Neste espaço de cores, a tonalidade de uma cor é representada pelo componente Matiz (H), que varia de 0 a 360 graus ao redor da roda de cores. A saturação (S) representa a pureza de uma cor e varia de 0 a 1, onde valores próximos a 0 representam tons de cinza e valores próximos a 1 representam cores mais saturadas. O valor (V) representa o brilho da cor, de 0 (preto) a 1 (cores mais claras) [45], [47].

Uma das principais vantagens do espaço de cores HSV é sua capacidade de separar informações de cor das informações de saturação e brilho, tornando-o adequado para muitas tarefas de processamento de imagem, como segmentação de objetos, imagem,

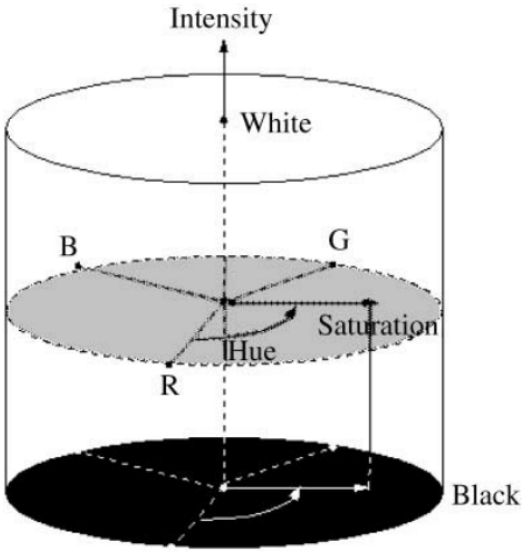


Figura 2.8: Representação do espaço de cores HSV [48]

saturação e brilho em HSV espacial, que podem ser executadas sem alterar o tom de cor, facilitando o reconhecimento e discriminação de objetos com base em suas características de cor [43].

Além disso, o espaço de cores HSV também é usado em aplicações de visão computacional, como reconhecimento de padrões e detecção de objetos, devido à sua capacidade de representar cores de uma forma assemelhada à percepção humana [49]. Esta representação mais intuitiva das cores facilita o desenvolvimento de algoritmos de processamento de imagens mais eficientes e fáceis de interpretar [50].

De acordo com Cheng (2000), as coordenadas podem ser transformadas utilizando o espaço de cores RGB por meio das seguintes fórmulas:

$$[ht]H = \begin{cases} 0^\circ & \text{se } V = 0 \\ 60^\circ \times \left(\frac{G-B}{V-\min(R,G,B)} \mod 6 \right) & \text{se } V = R \\ 60^\circ \times \left(\frac{B-R}{V-\min(R,G,B)} + 2 \right) & \text{se } V = G \\ 60^\circ \times \left(\frac{R-G}{V-\min(R,G,B)} + 4 \right) & \text{se } V = B \end{cases}$$

$$S = \begin{cases} 0 & \text{se } V = 0 \\ \frac{V-\min(R,G,B)}{V} & \text{caso contrário} \end{cases}$$

$$V = \max(R, G, B)$$

- **Matriz (H)**: Se a luminosidade (Valor) for 0, a matiz é definida como 0°. Caso contrário, é calculada com base no valor de cor correspondente ao elemento (R, G ou B) com o maior valor [48].

- **Saturação** (S): É baseado na diferença entre o valor máximo e mínimo dos componentes R, G e B, dividido pelo valor máximo encontrado [48].

- **Valor** (V): Este componente corresponde aos valores máximos entre cada os elementos R, G e B, ou seja, representado pela quantidade de brilho presente na cor [48].

RGB (Red, Green e Blue)

O espaço de cores RGB é um dos mais comuns de serem encontrados, sendo utilizado para representar as cores em dispositivos de exibição como monitores de computador, televisores e dispositivos móveis, bem como em softwares de edição de imagens e gráficos [44] (Figura 2.9). Neste contexto, cada cor é representada como uma combinação ponderada dos componentes vermelho (R), verde (G) e azul (B), que são os três componentes principais da família da luz secundária [47].

No espaço de cores RGB, cada componente varia de 0 a 255 (em um sistema de 8

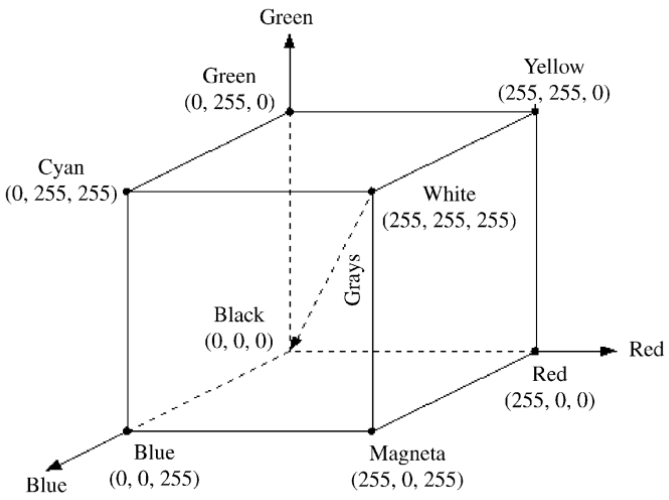


Figura 2.9: Representação do espaço de cores RGB [48]

bits), onde 0 representa nenhuma cor e 255 representa a intensidade máxima da cor. A combinação destas três cores primárias com intensidade variável permite a reprodução de uma ampla gama de cores visíveis ao olho humano [43]. Ele permite uma representação de cores precisa e flexível, o que é essencial em muitas aplicações, desde design gráfico até computação gráfica em jogos e filmes.

A conversão entre os espaços de cores RGB para HSV, pode ser realizada por meio da seguinte expressão matemática:

$$
\begin{aligned}
C &= V \times S \\
X &= C \times \left(1 - \left|\left(\frac{H}{60}\right) \mod 2 - 1\right|\right) \\
m &= V - C
\end{aligned}
$$

- H é a matiz (*Hue*) no intervalo de 0 a 360 graus,
- S é a saturação (*Saturation*) normalizada de 0 a 1,
- V é o valor (*Value*) normalizado de 0 a 1,
- C é a croma (*Chroma*),

- X é uma variável intermediária, e
- m é um fator de deslocamento.

Após calcular o valor de C e X, pode-se realizar o cálculo de conversão por meio da seguinte fórmula [44]:

$$(R,G,B) = \begin{cases} (C+m, X+m, m) & \text{se } 0 \leq H < 60 \\ (X+m, C+m, m) & \text{se } 60 \leq H < 120 \\ (m, C+m, X+m) & \text{se } 120 \leq H < 180 \\ (m, X+m, C+m) & \text{se } 180 \leq H < 240 \\ (X+m, m, C+m) & \text{se } 240 \leq H < 300 \\ (C+m, m, X+m) & \text{se } 300 \leq H < 360 \end{cases}$$

2.2.2 Cor e Textura

Para entender como funcionam a cor e a textura, precisamos entender como elas são percebidas na retina humana, que possui dois conectores principais: cor e haste. Primeiro, ao contrário das barras, necessárias para visualizar diferentes tons de cinza, as cores permitem a percepção das cores. Uma vez capturadas e armazenadas as imagens, é importante melhorar a sua qualidade sem perder propriedades essenciais. Para atingir esses objetivos, o uso de técnicas de processamento de imagens é essencial [51].

Um desses métodos é a segmentação, que divide uma imagem em múltiplas regiões com base em propriedades como cor, intensidade e textura [52]. A segmentação é particularmente útil para identificação de texturas porque destaca regiões de interesse com base em descontinuidades de contorno [53]. Esta abordagem permite uma análise mais detalhada da textura da imagem.

Maenpaa et al. (2004) propuseram dividir a análise de textura em dois métodos principais: processamento de cores e processamento de texturas, cada um deles associado a características específicas [53]. O objetivo desta análise é identificar imagens complexas

usando algoritmos matemáticos, examinando propriedades como brilho, cor e tamanho [53].

Rosenfeld (1976) também enfatizaram essa abordagem analítica, enfatizando que ela pode identificar imagens complexas e compreender sua composição visual. Utilizando técnicas avançadas de processamento de imagens, é possível interpretar texturas em termos de propriedades fundamentais, como a distribuição espacial de elementos visuais. [54].

Cor

A cor possui um papel importante sobre a vida do ser humano. Neste contexto, a cor pode ser utilizada para de diferentes formas, desde a expressão de algum sentimento e transmitir informações. Em meio ao mundo computacional, a física da luz e a psicologia da percepção visual são utilizadas para estudar tais comportamentos [47].

A luz visível é composta por diferentes comprimentos de onda, cada um correspondendo a uma cor específica do espectro eletromagnético [44]. Em uma pesquisa realizada por Szliski (2010), é possível visualizar de forma mais clara tais comportamentos, bem como as cores são influenciadas pelas interações entre a luz, os objetos e os receptores visuais do olho [42]. Existem muitos modelos e sistemas de cores, além daqueles já apresentados neste trabalho de dissertação, que podem ser utilizados em diferentes áreas [44].

Um modelo importante é o sistema CMYK utilizado na indústria gráfica para impressão colorida [42]. É utilizado em conjunto com o modelo RGB, que usa combinações de ciano, magenta e amarelo para criar uma variedade de cores. Existem também modelos de cores baseados em propriedades perceptivas, como o espaço de cores HSV [47].

A pesquisa realizada por Szliski (2010) aborda uma comparação entre duas operações aplicadas em diferentes escalas de cores. Na Figura 2.10, é realizada uma operação de subtração, que prepara a imagem para impressão. Esse processo é essencial para garantir que as cores originais sejam preservadas, mesmo após a conversão [42], [44].

Por outro lado, na Figura 2.11, ocorre um processo semelhante, no qual a imagem está sendo convertida da escala CMYK para RGB. Essa conversão demanda atenção devido às diferenças nos valores de gama e na reprodução das cores. É importante garantir uma

representação precisa das cores durante essa conversão, a fim de preservar a qualidade visual da imagem final [42], [44].

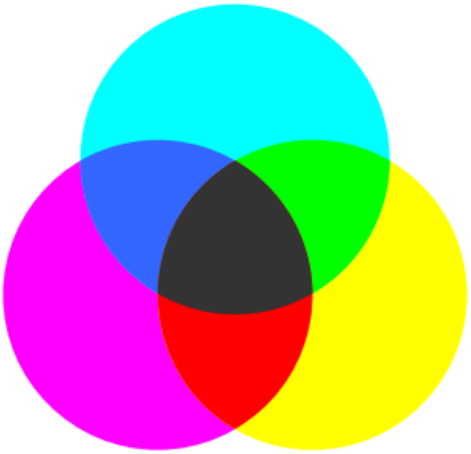


Figura 2.10: Imagem resultante da subtração da escala CMYK sobre RGB [42]

Figura 2.11: Imagem resultante da adição da escala RGB sobre CMYK [42]

Textura

O conceito do Histogram Equalization (HE) foi introduzido por W.K Pratt em 1968 [43], sendo utilizado como base para o desenvolvimento de diversos outros algoritmos de pré-processamento de imagem. A aplicação da técnica começa com o cálculo do histograma da imagem original, sendo uma representação gráfica da distribuição das intensidades de pixel na imagem [43]. Em seguida, uma função de transformação é calculada com base no histograma original. Essa função mapeia os valores de intensidade originais para novos valores de intensidade, de modo a equalizar o histograma [43].

Uma vez calculada a função de transformação, ela é aplicada a cada pixel na imagem original [49]. Isso é feito substituindo cada valor de intensidade original pelo valor correspondente na função de transformação. O resultado é uma nova imagem equalizada, na qual os pixels estão distribuídos de forma mais uniforme ao longo do intervalo de intensidades, resultando em um aumento geral no contraste e na qualidade visual da imagem [43].

O Global Histogram Equalization (GHE) utiliza uma transformação simples, no qual o histograma é calculado em toda a imagem de entrada e então passada para a etapa de cálculo de distribuição por meio de Cummulative Distribution Function (CDF). Na próxima etapa é realizada a derivação por meio de uma função de transferência cinza do CDF. Esse tipo de algoritmo, apesar de sua simples aplicação, pode resultar em perda de dados em regiões pequenas de uma imagem.

O Local Histogram Equalization (LHE) utiliza uma transformação mais localizada, sendo inicialmente obtido um histograma da região desejada para que então seja realizada a derivação por meio da função de transferência cinza do CDF. Logo após o pixel central da região é equalizado por meio desta função, isso fará com que a região retangular seja movida para o pixel adjacente, repetindo o processo de equalização. Esse método permite que se tenha um destaque maior sobre regiões de interesse, no entanto, possui complexidade elevada, tornando-o custoso em questões computacionais.

O Adaptive Histogram Equalization (AHE) foi proposto por Pizer pode ser descrito como sendo um método que visa a redução de complexidade computacional por meio de interpolação, de forma geral, esse método utiliza a transformação somente em regiões de amostra, e logo em seguida realiza a interpolação e transformação entre ambas as amostras locais (Figura 2.12).

O Contrast Limited Adaptive Histogram Equalization (CLAHE) é um método derivado do AHE, no entanto, utiliza como base um limite para a aplicação da transformação, para não comprometer a qualidade da imagem, e assim, obtendo um resultado eficiente tanto em aspectos computacionais, como também na imagem resultante.

2.3 Trabalhos Relacionados

Esta seção descreve tarefas relacionadas à classificação de imagens usando extração de camadas de modelos de aprendizado profundo. Neste contexto, descobri o trabalho que serviu de ponto de partida para este estudo. Andre Baloan e Nils Murrugarra da [55] introduziram a necessidade de um mecanismo de pesquisa mais prático que aumente o

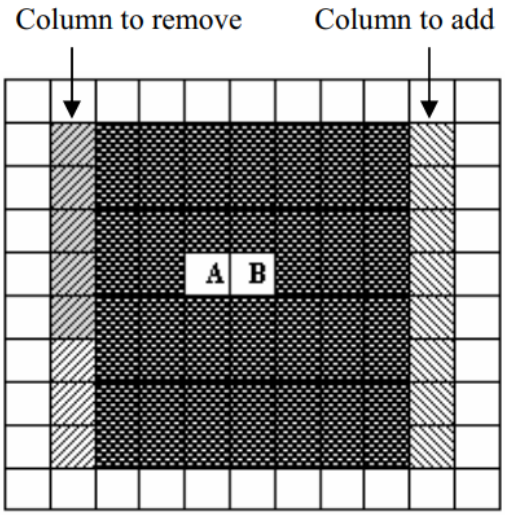


Figura 2.12: Funcionamento base do AHE

poder de pesquisa das consultas de texto, incluindo-se as consultas baseadas em imagens. O artigo completo apresenta abordagens para melhorar a qualidade das consultas online. Dessa forma, os autores propõem uma abordagem que utiliza modelos convolucionais, o que não é viável considerando a necessidade de reciclagem. Neste contexto, os autores propõem uma abordagem que utiliza camadas ocultas de uma rede convolucional para obter conhecimento de dados através da combinação de classificadores K-Nearest Neighbors (KNN). Ao longo do artigo, destaca-se como alternativa a possibilidade de implementação de blocos de redes neurais "ocultos" de forma não supervisionada. Nesse contexto, observou-se que determinados tipos de imagens tendem a ativar camadas mais profundas que nos ajudam a identificar melhor características como a cor, entre outras fontes.

Como metodologia, os autores utilizam a coleta de conjuntos de dados utilizando a plataforma Kaggle para coletar 100 imagens de duas categorias (cor e textura). A saída de cinco blocos residuais de um modelo ResNet50 treinado usando pesos ImageNet, especificamente os blocos residuais 2 e 5, foi usada para extração. O resultado consiste em um vetor de tamanho variável variando de 64 a 2048. A classificação foi realizada pelo método KNN utilizando espaço residual. Essa abordagem permite que novas características surjam sem treinamento adicional. Em vez disso, a classe representativa é comparada

com as classes vizinhas.

Capítulo 3

Desenvolvimento e Implementação

Este capítulo apresenta e descreve a metodologia seguida, bem como o processo de desenvolvimento do trabalho, constituído por três fases: construção do dataset de imagens, treino do modelo de classificação de imagens e avaliação dos resultados. Por fim serão apresentadas as ferramentas utilizadas, bem como a justificativa para o uso de cada uma.

3.1 Metodologia

A metodologia empregada neste estudo baseia-se em uma abordagem experimental, voltada para a investigação do impacto de variações de cor e textura nos diversos blocos residuais presentes no modelo de RNA, especificamente a ResNet50. O processo metodológico é dividido em várias etapas.

Inicialmente, o problema de classificação em questão é definido, identificando as características dos dados e as classes a serem previstas. Em seguida, os dados relevantes para o estudo são coletados e preparados para análise, incluindo limpeza, pré-processamento e transformação, garantindo que estejam em um formato adequado para o treinamento pelo modelo selecionado.

Após a coleta dos dados necessários, são gerados conjuntos utilizando diferentes alterações em suas escalas de cores e texturas, por meio de técnicas como ajuste na luminosidade, cor e transformações em seu contraste por meio da técnica de pré-processamento

CLAHE.

Em seguida, são selecionados diferentes blocos residuais, presentes na ResNet (já treinada com os pesos da ImageNet) para avaliação, considerando suas características e adequação ao problema em questão.

Os dados são então classificados utilizando KNN, variando os valores de vizinhos (k). Para isso, é empregada a validação cruzada para garantir a robustez dos resultados, prevenindo o *overfitting* e fornecendo uma estimativa mais precisa do desempenho dos modelos em dados não observados.

Por fim, os resultados são interpretados e analisados, tendo em consideração o desempenho dos diferentes algoritmos e sua aplicabilidade ao problema em questão. Na Figura 3.1, apresenta-se uma visão geral sobre a execução da metodologia abordada neste trabalho de dissertação.

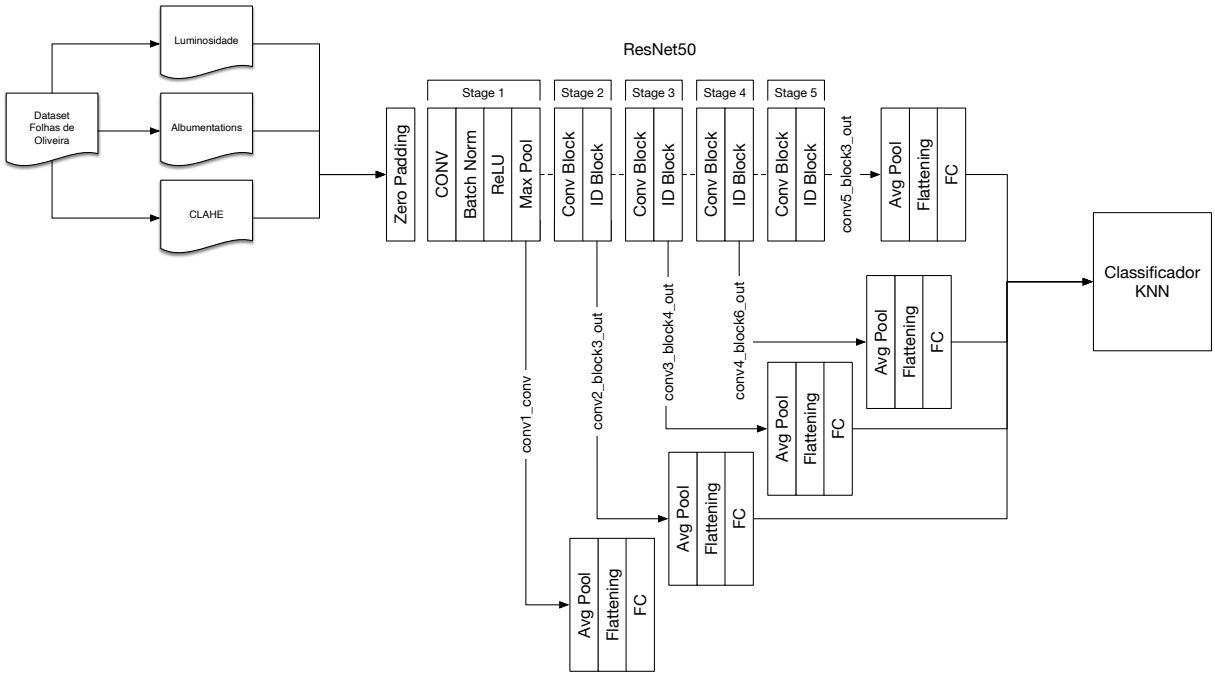


Figura 3.1: Esquema de execução da metodologia

3.2 Construção do Dataset

O conjunto de dados inicial consiste em uma coleção de imagens de folhas de oliveiras, contendo uma variedade de condições de saúde. Cada imagem foi tratada e filtrada para garantir o maior nível de detalhes e precisão análises futuras. As folhas foram classificadas em diferentes categorias de doenças, incluindo Carência de Boro, Olho de Pavão, Carência de Potássio e Folhas Saudáveis (Figura 3.2).

O conjunto de imagens foi introduzido no projeto de dissertação de Asseiro (2019) [56], com foco na pesquisa de doenças em folhas de oliveiras. Este projeto de dissertação utilizou de diversas estratégias de classificação para conseguir determinar por meio de uma aplicação móvel o tipo de doença para cada tipo de folha.

Na tabela 3.1 é possível visualizar os nomes atribuídos para cada uma das classes no conjunto.

Nome das Classes	**Descrição**
oli-bor-def	Carência de Boro
oli-heal	Folhas Saudáveis
oli-pea-spo	Olho de Pavão
oli-pot-def	Carência de Potássio

Tabela 3.1: Descrição das classes do conjunto de dados de folhas de oliveira.

As imagens iniciais apresentavam uma resolução de 256x256 pixels, tornando necessário realizar uma padronização, garantindo então que fossem aceitas pelas entradas do modelo selecionado.

Cada uma das classes contém uma variedade de até 169 amostras, todas utilizando o mesmo padrão de captura, sendo uma imagem com fundo branco e a folha de oliveira. O conjunto de imagens original possui um total de 676 imagens.

3.2.1 Pré-processamento

Durante a fase de pré-processamento, foram criados quatro conjunto de imagens, sendo estes apresentados na Tabela 3.1, para isso foram seguidos 4 passos importantes:

(a) Deficiência de Boro

(b) Saudável

(c) Olho de Pavão

(d) Deficiência de Potássio

Figura 3.2: Classes existentes no dataset

1. Coleta das imagens e suas classes originais.

2. Padronização das imagens, incluindo ajuste de tamanho e escalas de cores.

3. Seleção e análise da técnica de pré-processamento.

4. Aplicação da técnica ao conjunto de dados.

Na Figura 3.3, é possível visualizar um diagrama com todas as técnicas aplicadas sobre o conjunto de dados originais.

O processo de padronização de imagens foi realizado para garantir que todas as imagens estivessem em um tamanho adequado para o modelo. Neste contexto, foi aplicado um

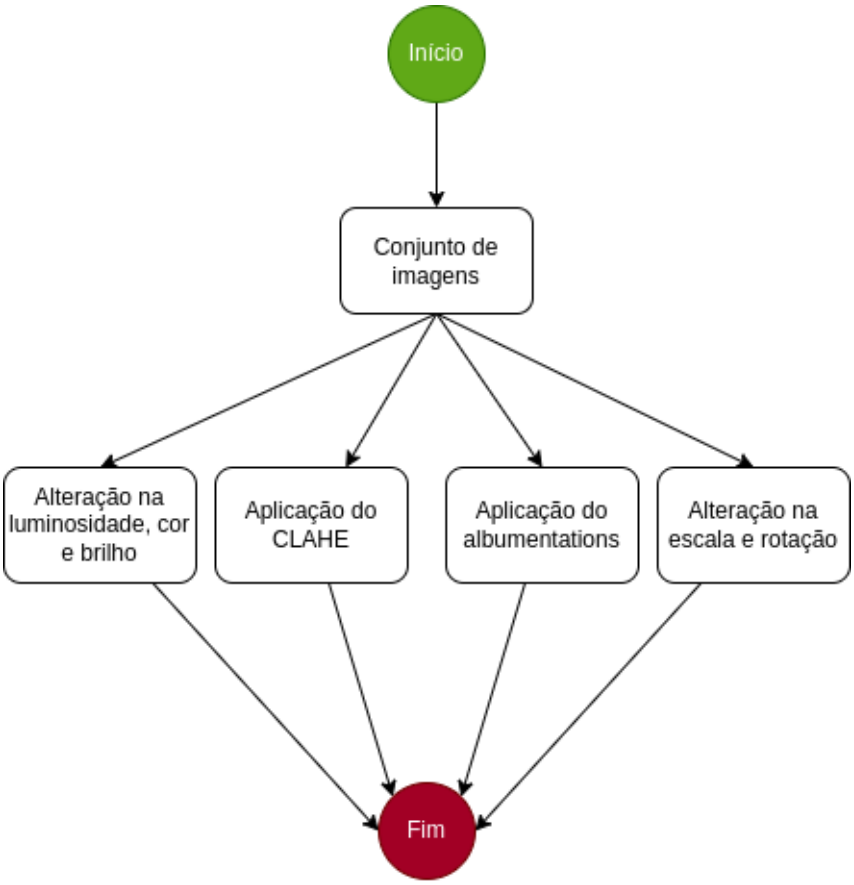


Figura 3.3: Diagrama de construção dos datasets

algoritmo sobre cada uma das imagens existente, permitindo então a mudança por meio da biblioteca OpenCV (`https://opencv.org/`) do tamanho 256x256 para 224x224. Na Figura 3.4 é possível visualizar um diagrama, mostrando os diferentes caminhos seguidos para implementar cada variação.

3.2.2 Selecionar as técnicas

Mediante ao modelo selecionado, e metodologia escolhida para esta pesquisa, notou-se uma fase importante, sendo necessário realizar uma análise para escolha das abordagens que seriam utilizadas para organizar os dados. O objetivo era agrupar as imagens conforme as variações e técnicas aplicadas, para isso, foi necessário entender a aplicação das alterações sobre cada uma das técnicas escolhidas.

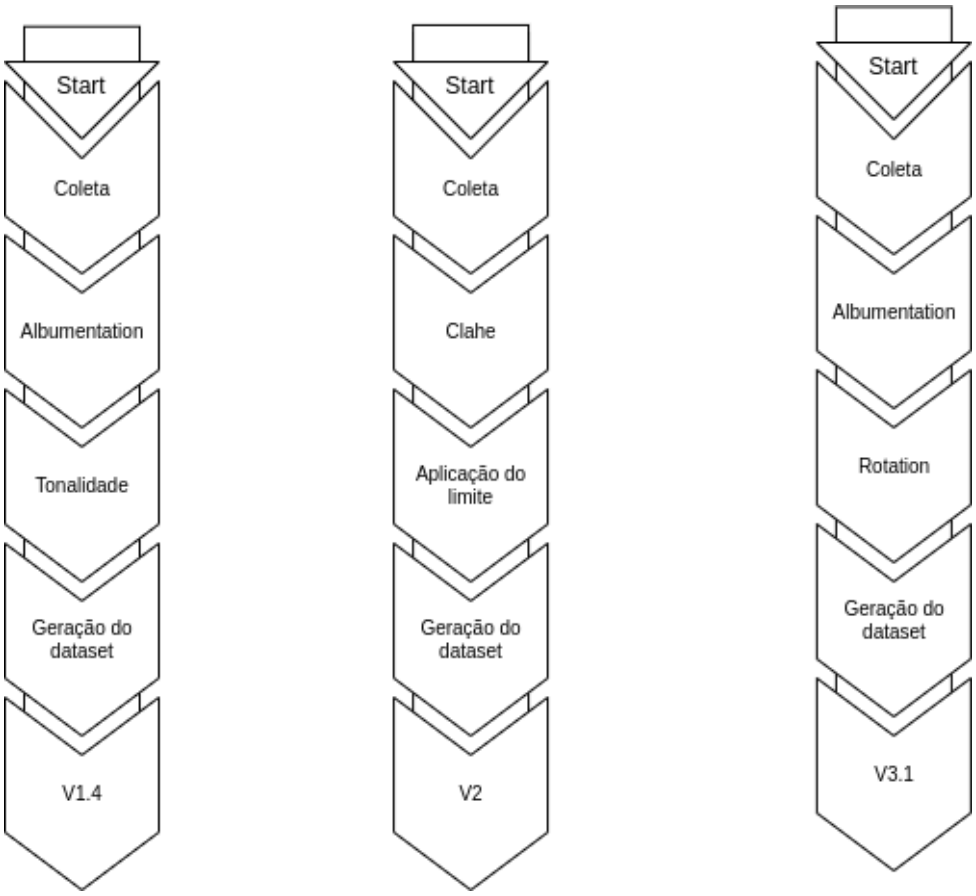


Figura 3.4: Construção de cada uma das classes

Neste contexto, notou-se a possibilidade de criar diferentes versões para cada uma das técnicas. Para melhor entendimento, é possível notar em meio a tabela 3.2 a primeira versão, do qual, foi utilizada uma técnica de ajuste de luminosidade, cor e também na saturação sobre o conjunto de imagens originais (sem ajuste nenhum). Essa abordagem permitiu obter uma visualização clara e objetiva sobre os novos conjunto de imagens gerados.

Identificador	**Categoria**
v1	Mudança na luminosidade, cor e saturação
v2	Aplicação de um limite no contraste
v3	Técnicas de Algmentation

Tabela 3.2: Técnicas aplicadas

Deste modo, cada alteração realizada a partir do dataset inicial, seria uma sub-versão

da categoria. Todas as alterações realizadas, foram implementadas a partir de bibliotecas oficiais e adequadas, dentre elas está a biblioteca Albumentation.

Por meio das abordagens introduzidas acima foi possível obter alterações significativas em cada classe, seja por meio do destaque da coloração, luminosidade, saturação ou até mesmo pela alteração do contraste.

Foram utilizadas duas bibliotecas fundamentais para a geração dos datasets, sendo a primeira o PILLOW e a biblioteca Albumentations permitindo assim a implementação dos filtros exigidos e também a separação entre as categorias desejadas, deste modo, está seção visa explicar melhor cada uma das técnicas, bem como apresentar os valores utilizados como parâmetro para cada dataset.

Alteração na Luminosidade, Cor e Saturação

A primeira técnica utilizou-se da biblioteca PILLOW em meio a linguagem Python, visando buscar um conjunto de valores que juntos melhorassem a imagem, tanto em efeitos de iluminação, como também em cor e saturação de uma imagem, abaixo estão descritos cada ajuste:

- **Luminosidade**: É utilizada para melhorar a percepção visual, envolvendo alterações no brilho da imagem por completo.

- **Saturação**: É utilizada para referenciar-se ao ajuste na intensidade das cores impostas na imagem, isso possibilita aprimorar as cores vibrantes da imagem, como também criar tons de cinza para gerar imagens em escala de cinza.

- **Cor**: Refere-se ao ajuste em meio aos canais de cores existentes em uma imagem, tendo como base a escala mais conhecida RGB (Red, Green, Blue).

Na tabela 3.3 é possível visualizar o conjunto de valores utilizado para cada um dos ajustes.

Versão	Função aplicada	Valor
v1.1	ImageEnhance.Brightness	0.5
v1.2	ImageEnhance.Color	1.3
v1.3	ImageEnhance.Contrast	1

Tabela 3.3: Funções aplicadas para a versão 1

Aplicação do Limite com o CLAHE

Essa técnica foi utilizada no pré-processamento de imagens para aprimorar detalhes que pudessem aprimorar detalhes em áreas de baixo contraste. Essa abordagem utiliza um limite imposto antes da aplicação do filtro, e pode ser aplicadas em diversas outras categorias de imagens, seja exames laboratoriais, como também em fotografias distintas. Para a aplicação desta técnica foram realizadas análises que em colaboraram com o objetivo deste trabalho, justamente pelo fator de ser utilizada para melhorar características como texturas em imagens. Neste contexto, o CLAHE utiliza uma sequência de passos, dos quais serão descritos na figura 3.5:

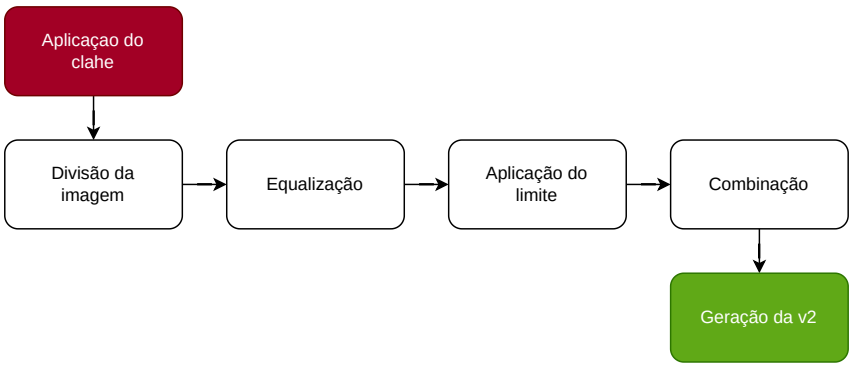


Figura 3.5: Execução do CLAHE

- **Divisão da imagem**: divide a imagem em pequenas regiões, também chamadas de *blocos*, representando um local da imagem.
- **Equalização de histograma local**: para cada bloco encontrado, é realizado um processamento de distribuição de intensidade, isso irá mostrar a frequência de ocorrência sobre diferentes valores de intensidade sobre aquele bloco.

- **Limite do contraste**: um dos principais passos desta técnica é a aplicação de um limite, do qual, é utilizado sobre o contraste, evitando que o processo de equalização amplifique demais o contraste e torne o processo custoso para o computador, e cause efeitos indesejados sobre o resultado.

- **Junção dos blocos**: após realizar o processamento de equalização sobre as regiões, é realizado um processo de combinação e gerado a imagem final com o contraste melhorado.

Neste conceito, o CLAHE foi implementado utilizando a biblioteca do próprio OpenCV para a linguagem de programação Python, e teve como base os valores descritos na tabela 3.4.

Versão	**Função aplicada**	**Valor do limite**	**Tamanho do grid**
v2.1	cv.createCLAHE	5	[8,8]

Tabela 3.4: Funções aplicadas para a versão 2

Aplicação do Albumentation

Nesta etapa foi utilizada a biblioteca do Albumentations para a aplicação, deste modo, permitiu trabalhar com uma biblioteca de alto nível de código, proporcionando flexibilidade e eficiência durante a implementação. O principal objetivo desta implementação se deve ao fato de conseguir ampliar realizar operações na imagem para conseguir verificar o processo de identificação da folha, e medir o comportamento das classes mediante a alterações simples como: rotação e escala de uma imagem.

- **Rotação**: a transformação tinha como objetivo criar um dataset com diferentes ângulos de visualização da imagem, permitindo que o classificador consiga reconhecer padrões distintos.

- **Flip Horizontal**: utilizada para verificar em conjunto com a rotação, se o modo consegue identificar apesar da orientação do objeto.

Neste contexto, não se quis utilizar imagens sintéticas para o treino, justamente pelo fator de conseguir analisar com precisão o comportamento das camadas mais profundas em relação a diferentes tipos de alterações que podem ocorrer durante a classificação, e então conseguir medir o nível de sensibilidade em relação a alterações como cores e texturas, na tabela 3.5 é possível visualizar os valores utilizados para este teste, onde a rotação será representada por (Rot) e o Horizontal Flip por (Flip).

Versão	**Função aplicada**	**Valores (Rot)**	**Valores (Flip)**
v3.1	A.Rotate	min=-35 e max=35	0
v3.2	A.HorizontalFlip, A.Rotate	min=-35, max=35	0.2
v3.3	A.HorizontalFlip, A.Rotate	min=-15, max=15	0.2
v3.4	A.Rotate	min=-15, max=15	0

Tabela 3.5: Funções aplicadas para a versão 3

3.3 Treino e Execução dos Modelos

O processo de treinamento foi realizado mediante a cada um dos conjuntos de dados gerados, sendo para cada um dos blocos residuais da ResNet50. O modelo escolhido, foi utilizado em conjunto com os pesos da ImageNet, permitindo que o modelo inicialize com os pesos pré-treinados. A implementação foi realizada por meio da biblioteca do TensorFlow (`https://www.tensorflow.org/?hl=pt-br`).

Toda a implementação foi realizada em dois ambientes distintos, ambos empregando uma placa de vídeo dedicada para os testes. As configurações desses ambientes podem ser encontradas na Tabela 3.6.

Local	**Tipo da placa**	**Quantidade de RAM**
Máquina física	GPU	32 GB
Google Colab	GPU	20 GB

Tabela 3.6: Configuração do ambiente

Como mencionado no início desta dissertação, o objetivo é avaliar a sensibilidade mediante a diferentes alterações no conjunto de dados, aplicando sobre cada um dos

blocos existentes no modelo ResNet50. Nesse contexto, foi desenvolvido um algoritmo para representar a estratégia desejada de treinamento, seguindo uma sequência de passos desde a aquisição do conjunto de dados, o treinamento e, posteriormente, a geração das métricas de avaliação. O algoritmo segue os seguintes passos:

1. **Get data**: Recebimento do conjunto de informações
2. **Get layer**: Extração da camada desejada
3. **Encode**: Codificação dos dados
4. **Split**: Seração das informações
5. **Train**: Treinamento da camada
6. **Evaluate**: Geração das métricas

3.3.1 Extração das camadas

O algoritmo inicia o processo de treinamento, realizando a captura das camadas desejadas, que podem ser visualizadas na tabela 3.7.

Nome	**Saída**
conv1_conv	(112, 112, 64)
conv2_block3_out	(56, 56, 256)
conv3_block4_out	(28, 28, 512)
conv4_block6_out	(14, 14, 1024)
conv5_block3_out	(7, 7, 2048)

Tabela 3.7: Camadas extraídas

O processo de captura se baseia na captura das camadas de ativação para cada bloco de convolução existente na ResNet50, o processo inicia por meio da instância do modelo por meio da função *tf.keras.application.ResNet50* utilizando os pesos da ImageNet e com uma quantidade de 1000 classes. Logo em seguida, a camada extraída passa por um processo de diminuição de dimensionalidade do mapa de características, por meio da chamada do método *tf.keras.layers.GlobalAveragePooling2D*, esse processo torna-rá possível

a captura de *features* importantes para processos seguintes. Ao final do processo a saída será instanciada como um modelo por meio da função *tf.keras.Model* utilizando a camada de entrada inicial, e também a saída do Pooling.

Em seguida o algoritmo recebe o indicador da versão ao qual desejaria realizar os testes, deste modo o algoritmo iria realizar o carregamento das classes existentes na versão, e passando para a fase de pré-processamento, ao qual as imagens eram convertidas para o sistema de cores RGB (utilizando os três canais de cores), e convertidas para o tamanho 224x224 (garantindo que todas as imagens estejam no formato aceito pelo modelo escolhido).

A transformação dos dados após o processo de recebimento, garante que os rótulos para cada imagem existente, consiga ser compreendido pelo modelo de entrada, esse processo é realizado por meio da biblioteca Scikit-Learn, do qual permite utilizar diversos pré-processadores, bem como a função *fitTransform* do módulo *LabelEncode*.

Assim como em outros treinamentos, o processo de *split* também é necessário, deste modo, foi utilizada a função de *train_test_split* da biblioteca *Scikit-Learn* resultando na seguinte separação: o conjunto foi separado em 70% para o treinamento e 30% para o teste.

Ao final deste processo, o treinamento é finalmente inicializado e para isso foi utilizado o classificador *KNeighborsClassifier* utilizando uma quantidade de vizinhos instanciado no início da execução. Neste contexto, foi utilizado um range de 10 para a quantidade de vizinhos conhecidos pelo KNN, e com isso para cada quantidade de vizinhos o processo iria se repetir. O processo da classificação é então realizado por meio da chamada do método *knn.fit* passando o conjunto de treinamento separado anteriormente.

Logo após é inicializado o processo de coleta das métricas (Acurácia e Erro) durante o processo de classificação, na tabela 3.8 é possível visualizar as funções utilizadas para cada métrica.

Métrica	Função
Acurácia	accuracy_score
Erro	log_loss

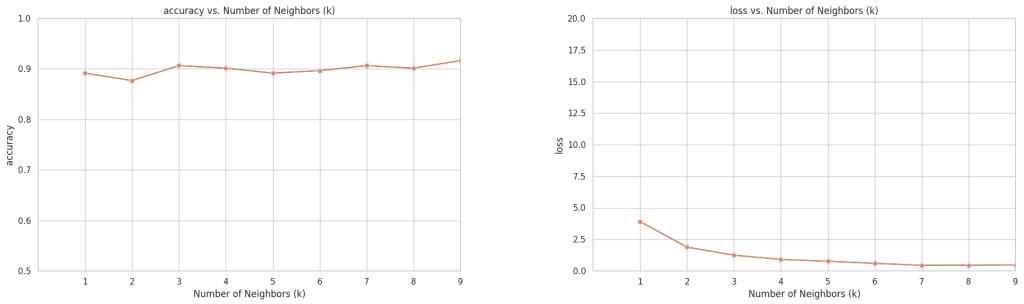
Tabela 3.8: Métricas utilizadas

3.3.2 Resultados

Após a conclusão do treinamento do modelo, iniciou-se à geração dos gráficos para cada versão considerada inicialmente. Esses gráficos ilustram o desempenho do classificador em relação a diferentes números de vizinhos.

Para cada camada de ativação capturada durante o processo, os gráficos correspondentes foram gerados, mostrando a taxa de acerto e a taxa de erro em relação ao número de vizinhos considerados. Essas visualizações fornecem análises sobre como o desempenho do modelo varia conforme a quantidade de vizinhos utilizados no algoritmo de classificação.

Os gráficos de taxa de acerto mostram a porcentagem de classificações corretas em relação ao número de vizinhos, enquanto os gráficos de taxa de erro mostram a porcentagem de classificações incorretas. Essas informações são essenciais para entender a eficácia do modelo em diferentes cenários e podem orientar ajustes e otimizações futuras.

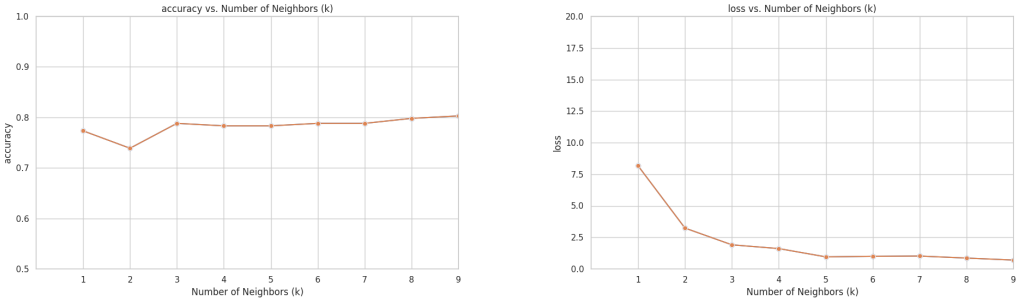


(a) Acurácia da quarta layer (b) Perda da quarta layer

Figura 3.6: Resultados da quarta layer do 1 dataset

Na Figura 3.6a, é possível visualizar o comportamento do modelo durante a classificação utilizando o conjunto de dados com alterações apenas no brilho. Sendo assim, o modelo alcançou uma taxa de acerto de 91,62%, representando um desempenho consideravelmente bom. No entanto, a máxima eficácia foi observada ao adotar sete vizinhos

para o processo de classificação. Esse resultado destaca a sensibilidade do modelo às variações na luminosidade das imagens, evidenciando a importância de considerar tais ajustes durante a análise e classificação das imagens.



(a) Acurácia da quarta layer (b) Perda da quarta layer

Figura 3.7: Resultados da quarta layer do 2 dataset

Na Figura 3.7, aplicamos o conjunto de dados com alterações no limite do contraste, destacando a sensibilidade da primeira camada às mudanças de contraste. Isso sugere que as camadas superficiais da ResNet são mais suscetíveis a alterações na cor e no contraste das imagens.

A tabela 3.9, apresentada os resultados para cada camada de treinamento, permitindo comparações entre as diferentes alterações aplicadas.

Versão	**Alteração**	**Melhor camada**	**Quant. K**
v1.1	Brilho	4	k=9
v1.2	Cor	5	k=9
v1.3	Contraste	3	k=6
v2.1	Contraste com limite	5	k=4
v3.1	Rotação [-35,35]	5	k=7
v3.2	Rotação e Flip [-35,35, 0.2]	5	k=7
v3.3	Rotação e Flip [-15,15, 0.2]	5	k=4
v3.4	Rotação [-15,15]	5	k=7

Tabela 3.9: Configuração utilizada

3.4 Ferramentas e Linguagens de programação

Nesta secção será realizada a apresentação das ferramentas e linguagens as quais foram utilizadas para o desenvolvimento do mesmo, bem como apresentação das bibliotecas e demais acessórios.

3.4.1 Python

Desenvolvida no início da década de 1990 por Guido Van Rossum por meio da Fundação Centro de Matemática (CWI) nos Países baixo, teve como antecessor a linguagem ABC. O Python é baseado em orientação a objetos, possui estrutura funcional, além de ser categorizada como uma linguagem de alto nível, tornando-á mais fácil para entender [57]. Existem diversas versões disponíveis da linguagem, além de um amplo sistema de contribuição de código, podendo ser aplicada a diversas áreas do conhecimento, bem como Inteligência Artificial, Análise de dados, desenvolvimento de aplicações para à internet quanto Graphical User Interface (GUI).

Abaixo estão descritas alguma das vantagens de se utilizar o Python:

- Múltiplas Threads: Permite ao desenvolvedor criar programas que não possuem dependência sequencial, deixando que a mesma seja executada de forma secundária, podendo aplicar diferentes tipos de controle, ao fim da execução [57].

- Desenvolvimento em roteiros: A linguagem permite trabalhar de forma simplificada com funcionalidades do sistema operacional, podendo realizar manipulações de arquivos, processos, dentre outros, de forma simplificada e sem muita complexidade [58].

- Controle de qualidade: O Python também possui um amplo sistema de controle de qualidade, permitindo realizar a criação de testes para o código gerado, podendo monitorar com facilidade as funcionalidades criadas e também possíveis erros, falhas e bugs que podem ocorrer após serem disponibilizado para o cliente [57][58].

- Aplicação de Padrões: Assim como em outras linguagens de programação, o Python também possuí um módulo de identificação de expressões regulares, podendo identificar, manipular, e até mesmo otimizar soluções mais complexas [57][58].

O Python utiliza um sistema de compilação de código de 3 três fases, iniciando com a geração do código-fonte, contendo as instruções desejadas para aquela determinada rotina, tendo isso em vista, o sistema irá passar para a geração do *byte code* que será gerado durante o processo de compilação, e enviado para a última etapa do processo, sendo à execução pela Máquina Virtual Python (PVM) responsável por executar o código já compilado, bem como é possível visualizar na figura 3.8.

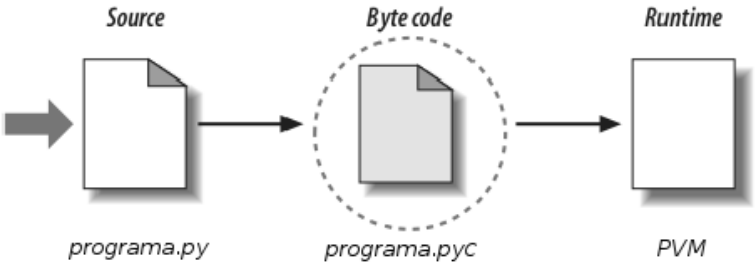


Figura 3.8: Processo de compilação Runtime Python [58]

3.4.2 Tensorflow

O TensorFlow é uma biblioteca de código aberto desenvolvida pela equipe de desenvolvimento do Google Brain que permite utilizar técnicas avançadas de Machine Learning e Deep Learning. Permite também o uso por meio de ambientes virtuais, como o Google Colab que disponibiliza opções variadas de processamento, bem como via tensores quanto processamento gráfico. Este ambiente permite a partilha do código existente, bem como controle de versão interno [59][60].

3.4.3 Keras

O Keras é uma API de alto nível e fornecida por meio da biblioteca TensorFlow, que permite realizar diversas operações, bem como selecionar arquiteturas já prontas, ou até

mesmo criar um modelo do zero, seguindo um padrão de criação desejado, adicionando ou removendo camadas, modificando os optimizadores, podendo também alterar os parâmetros de entrada e saída do modelo [61][60].

3.4.4 Sklearn

O Scikit-Learn é uma biblioteca de aprendizado de máquina (supervisionado e não-supervisionado) e também de código aberto que fornece ferramentas eficazes para análise preditiva de dados, possuindo compatibilidade com bibliotecas, como NumPy e SciPy. Por meio desta biblioteca é possível trabalhar com algoritmos de classificação, regressão e clusterização de informações [62][60].

Seleção de modelos

O módulo de seleção possui uma vasta quantidade de ferramentas para avaliação e seleção de modelos, sendo úteis para otimizar parâmetros, realizar validação e também separar o conjunto de treinamento.

Pré-processadores

Este módulo permite realizar um tratamento nos dados, antes mesmo de serem inseridos no modelo para o treinamento, isso permite padronizar as informações, dentre os mais conhecidos, se destacam o *StandardScaler* utilizado para dados contínuos e numéricos, tendo como base o uso de um algoritmo de aprendizado sensível à escala de recursos [63][60]. Outro pré-processador existente em meio a esses recursos, é o *LabelEncoder* utilizado para transformar variáveis categóricas em valores inteiros, permitindo ter uma visualização mais clara sobre as classes e também possibilitando o aprendizado [64].

3.4.5 OpenCV

A biblioteca do OpenCV foi desenvolvida pela Intel em 2000, com foco no desenvolvimento de visão computacional e de código aberto, que pode ser integrada a linguagens

como C++, Python, Java e MATLAB. Foi projetada para fornecer uma vasta quantidade de ferramentas e algoritmos que podem auxiliar na manipulação de imagens, análise e processamento de informações, entre outras. Também pode ser utilizada para aplicar técnicas avançadas sobre imagens, como reconhecimento, rastreamento de objetos, aplicação de algoritmos, dentre outros recursos [65].

3.4.6 Albumentation

A biblioteca do Albumentation foi desenvolvida em 2018 por um grupo de pesquisadores e engenheiros, como: Alexander Buslaev, Vladimir Iglovikov e Alex Parinov, para flexibilizar a eficiência dos conjuntos de dados na área de visão computacional. Essa técnica permite ampliar o dataset significativamente, possibilitando realizar manipulações de escala, rotação, translação, ajuste de cor, dentre outras modificações. Permite trabalhar com diferentes escalas de imagens (RGB, Grayscale, Multi-Espectral, entre outras) e pode ser utilizado em conjunto com bibliotecas de alto nível, como TensorFlow e PyTorch, abaixo podem ser visualizados alguns exemplos práticos utilizando a biblioteca em conjunto com a versão 3.8 do Python.

Capítulo 4

Resultados e Discussão

Este capítulo apresenta e analisa os resultados relativos à classificação da sensibilidade de cada blocos residuais da ResNet50 para as variações de cor e textura, utilizando o classificador KNN.

4.1 Testes utilizando o classificador KNN

Foi implementado o algoritmo de classificação KNN para medir o impacto nas diferentes camadas da ResNet50. Para isso, foi realizada uma série de testes utilizando diferentes números de vizinhos, variando de 1 a 10, para cada um dos conjuntos de dados gerados na etapa de pré-processamento.

O processo iniciou com a coleta das camadas de saída correspondentes a cada um dos blocos residuais, conforme detalhado na Tabela 3.7.

Este procedimento não apenas permitiu explorar a capacidade do classificador K-nearest neighbors em distinguir entre os diferentes padrões presentes nos dados, mas também avaliar a sensibilidade de cada camada em relação às diversas versões geradas durante a etapa de pré-processamento.

4.2 Análise das versões

Para conduzir esta análise, serão consideradas as matrizes de confusões e também os gráficos de desempenho (acurácia e perda) geradas em cada etapa de classificação. Esses gráficos fornecerão uma visualização clara e objetiva da relação entre os diferentes valores de K mencionados anteriormente neste capítulo, permitindo uma avaliação precisa do desempenho do modelo ao final do processo de classificação, descritas abaixo.

1. Mudança na luminosidade, cor e saturação: Ajuste na aparência visual das imagens, realçando ou suavizando luminosidade, cor e saturação.

2. Aplicação de um limite no contraste: Ajuste no contraste por meio de um limite, garantindo que as características sejam mantidas, de forma, a destacar os detalhes sem manter a qualidade visual.

3. Técnicas de algmentation: Ajuste por meio da rotação e *Horizontal Fip* nas imagens, obtendo diversos ângulos diferentes para avaliação.

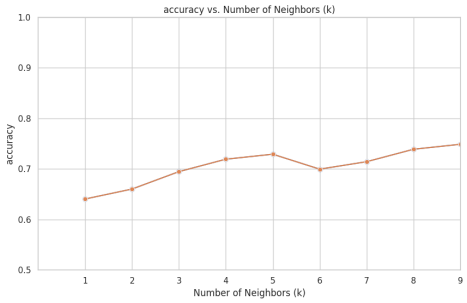
4.2.1 Mudança na luminosidade

A primeira versão é composta por 3 ajustes, podendo ser conferido na tabela 3.3. Após realizar uma análise sobre cada um dos blocos residuais, para cada um dos ajustes, notou-se que em determinadas situações, camadas distintas conseguiram ser mais sensíveis mediante aos ajustes.

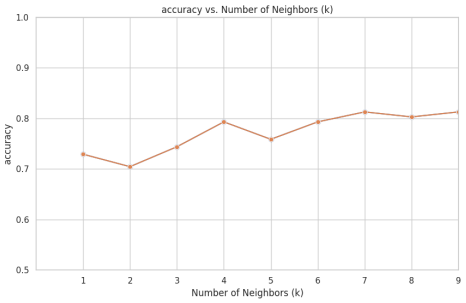
Brilho

Relativamente à análise realizada após a classificação utilizando variações no brilho das imagens, notam-se pontos de alta em cada um dos processos de classificação. Neste contexto, nota-se que a sensibilidade referente ao ajuste realizado, foi alcançado em momentos diversificados ao longo do processo de classificação.

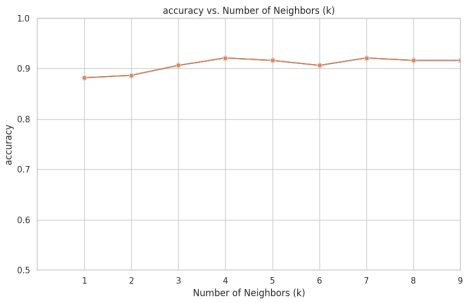
Ao realizar essa análise, nota-se por meio do gráfico de acurácia, que existe uma semelhança entre os blocos residuais 3 e 4, no qual atingiram seu ponto de máxima por



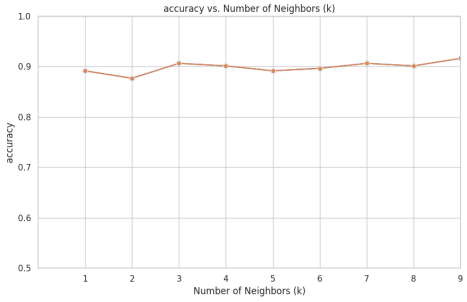
(a) *conv1_conv*



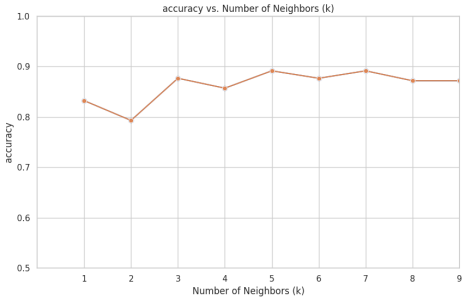
(b) *conv2_block3_out*



(c) *conv3_block4_out*



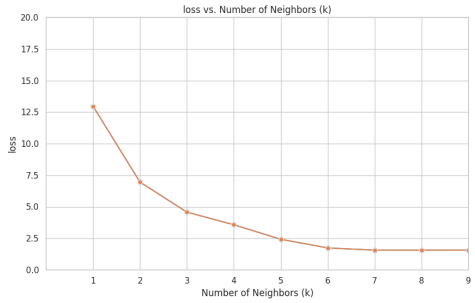
(d) *conv4_block6_out*



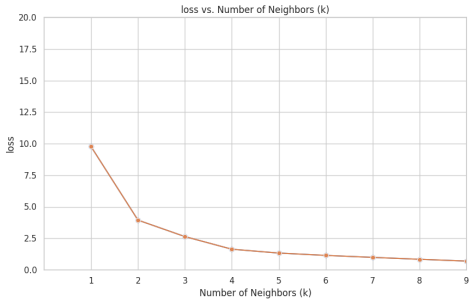
(e) *conv5_block3_out*

Figura 4.1: Gráficos de acurácia para a versão 1.1

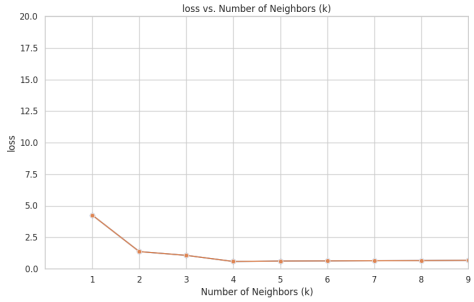
meio do valor de (k=4,7 e 9). Os demais pontos de máximas podem ser visualizados por meio da tabela 4.1



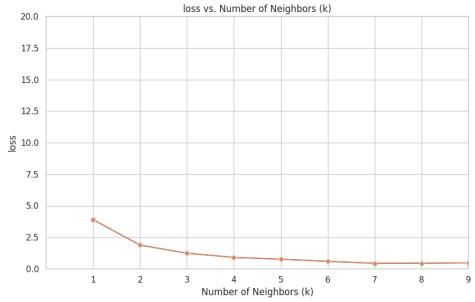
(a) *conv1_conv*



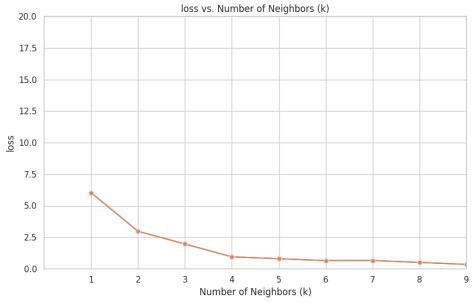
(b) *conv2_block3_out*



(c) *conv3_block4_out*



(d) *conv4_block6_out*



(e) *conv5_block3_out*

Figura 4.2: Gráficos de perda para o ajuste no brilho

Em meio a análise realizada, nota-se que em meio aos gráficos de perda, também há diferentes momentos em que o bloco residual atinge o menor percentual de perda. Neste contexto, é possível visualizar que a camada 4 em conjunto com o valor de vizinhos (k=7) possuí o melhor desempenho, tendo o valor de acurácia (sendo o maior dentre os testes).

Camada	Acurácia final	Maior acurácia	Valor k	Perda final
conv1_conv	0.749	0.749	k=9	1.567
conv2_block3_out	0.812	0.812	k=4,9	0.691
conv3_block4_out	0.916	0.924	k=4,7	0.663
conv4_block6_out	0.916	0.921	k=9	0.479
conv5_block3_out	0.871	0.869	k=5	0.3573

Tabela 4.1: Acurácia das camadas para o ajuste no brilho

No entanto, se observado a figura 4.2 de perda e a tabela 4.1, nota-se uma variação entre ambas as camadas.

A figura 4.3 permite visualizar a distribuição de classes para a alteração do brilho

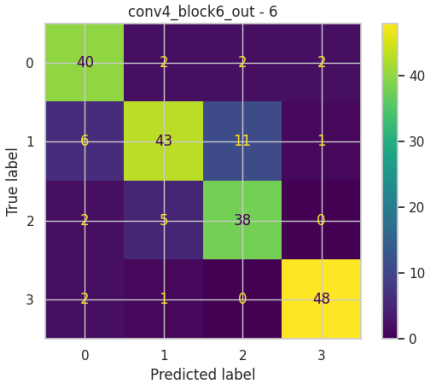


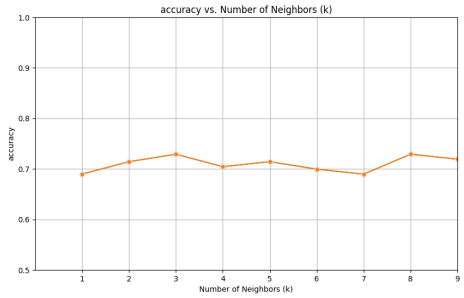
Figura 4.3: Matriz de confusão utilizando da 4 layer k=7

Cor

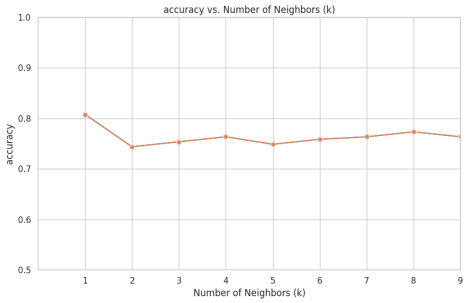
Após realizar a alteração na cor das imagens, notou-se uma mudança significativa nos blocos mais sensíveis, sendo destacado o quinto bloco residual.

Para este ajuste, foi alcançada uma acurácia final de 95,0%. Este valor foi alcançado mediante ao uso do número de (k=9), sendo a primeira camada como a que obteve o melhor desempenho, se comparado às outras camadas. Nota-se uma variação de acurácia, quando comparado as camadas mais profundas, obtendo valores bem próximos.

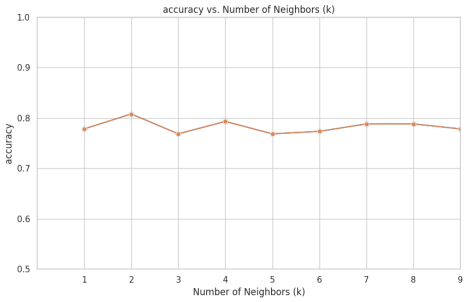
Quando comparado o valor de perda para este ajuste, nota-se um aprofundamento das camadas, obtendo um valor de acurácia por meio da segunda camada *conv5_block3_out*,



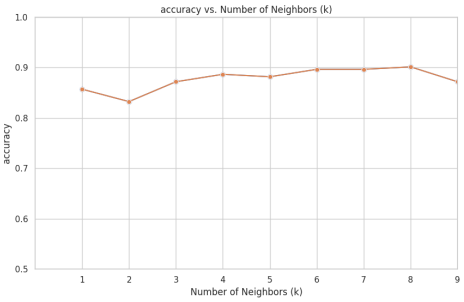
(a) *conv1_conv*



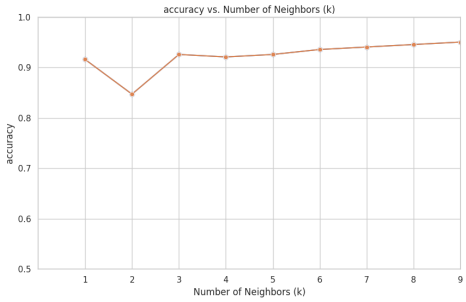
(b) *conv2_block3_out*



(c) *conv3_block4_out*



(d) *conv4_block6_out*

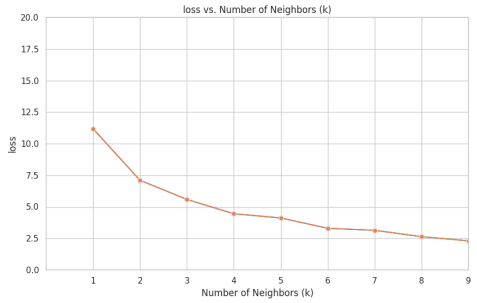


(e) *conv5_block3_out*

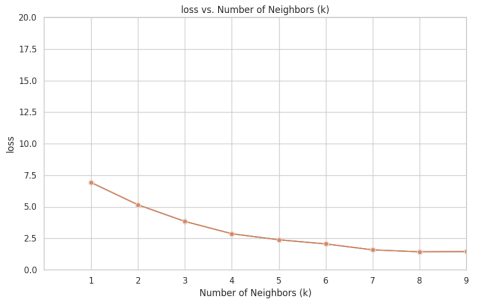
Figura 4.4: Gráficos de acurácia para o ajuste na cor

Camada	**Acurácia final**	**Maior acurácia**	**Valor k**	**Perda final**
conv1_conv	0.719	0.741	k=8	2.302
conv2_block3_out	0.763	0.803	k=1	1.403
conv3_block4_out	0.778	0.809	k=1	0.909
conv4_block6_out	0.871	0.901	k=8	1.067
conv5_block3_out	0.950	0.950	k=9	0.405

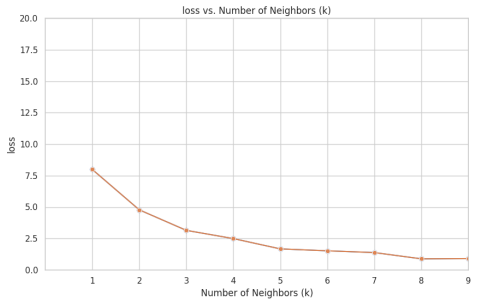
Tabela 4.2: Acurácia das camadas para o ajuste na cor



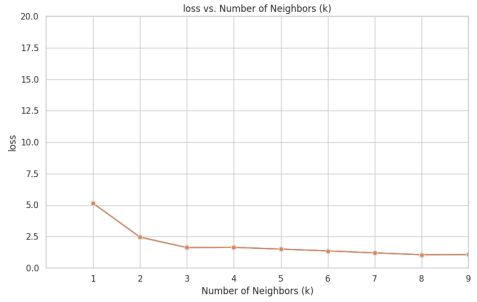
(a) *conv1_conv*



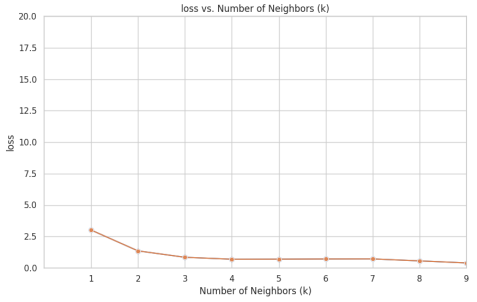
(b) *conv2_block3_out*



(c) *conv3_block4_out*



(d) *conv4_block6_out*



(e) *conv5_block3_out*

Figura 4.5: Gráficos de perda para o ajuste na cor

atingindo um percentual de 0.951, como é apresentado na tabela 4.2.

Na figura 4.5 é possível visualizar o comportamento da perda em relação ao valor do número de vizinhos utilizados, desta forma, nota-se uma variação muito pequena em meio ao valor com a melhor acurácia. No entanto,

A figura 4.6 permite visualizar a distribuição de classes para a alteração de cor

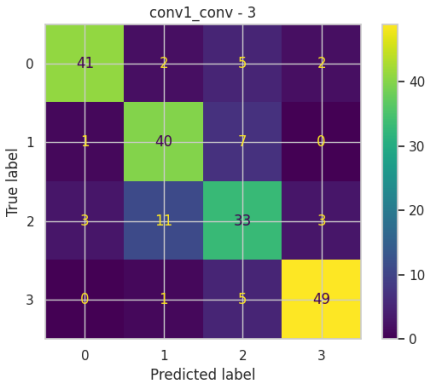


Figura 4.6: Matriz de confusão utilizando k=4

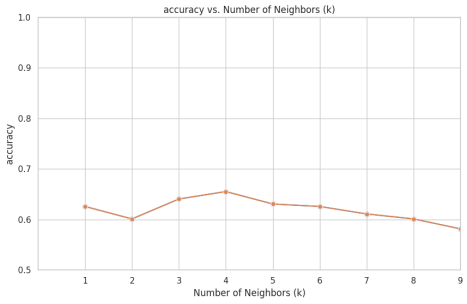
Contraste

Durante a análise de contraste, notou-se uma mudança significativa entre os blocos residuais mais sensíveis. Especificamente, nas camadas mais profundas, tendo melhor desempenho nesse aspecto.

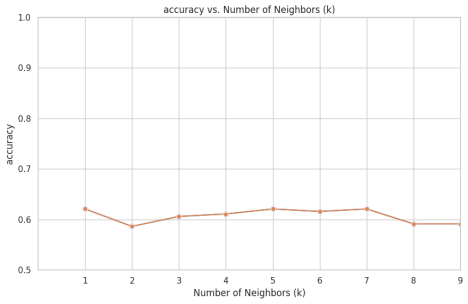
O ajuste do contraste das imagens, permite realçar os padrões complexos, facilitando assim o reconhecimento de características sutis ou detalhes importantes. Essa capacidade de destacar padrões complexos torna o ajuste de contraste uma técnica valiosa em problemas de visão computacional, como a detecção de bordas, a segmentação de objetos e outras tarefas que exigem uma compreensão detalhada da estrutura das imagens.

Desta forma, o valor final de acurácia para este ajuste foi de 76.8%, sendo o uso do número de vizinhos (k) igual a 6,7 e 8 para a camada 3 e 4, como mostra a figura 4.7.

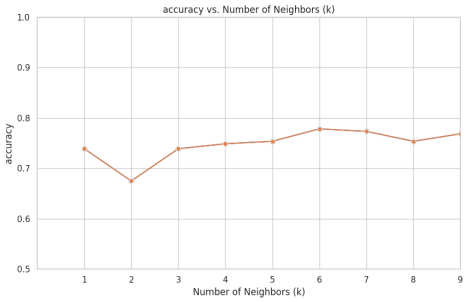
É importante ressaltar que as camadas mais profundas, destacam-se pela identificação de texturas, ou seja, são sensíveis às características mais complexas de uma imagem.



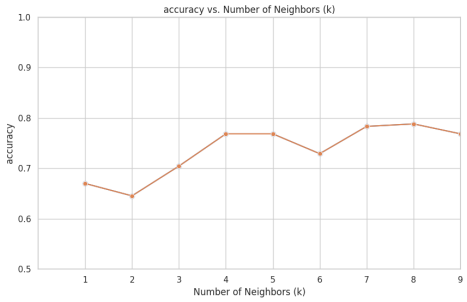
(a) conv1_conv



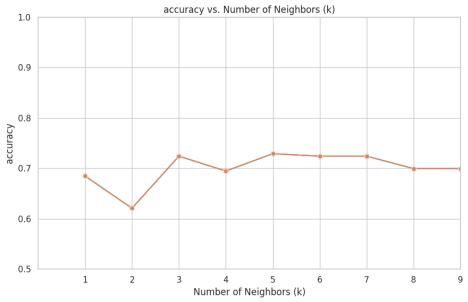
(b) *conv2_block3_out*



(c) *conv3_block4_out*



(d) *conv4_block6_out*

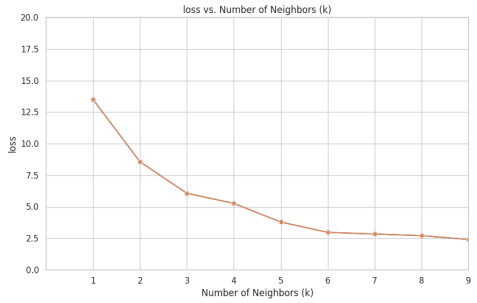


(e) *conv5_block3_out*

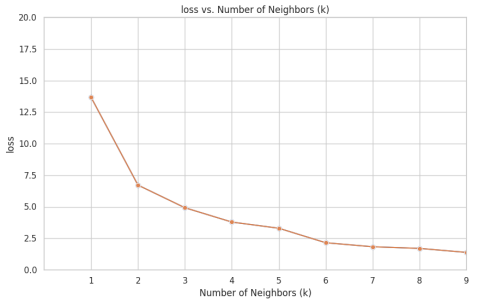
Figura 4.7: Gráficos de acurácia para o ajuste no contraste

Camada	**Acurácia final**	**Maior acurácia**	**Valor k**	**Perda final**
conv1_conv	0.581	0.678	k=4	2.411
conv2_block3_out	0.591	0.634	k=5	1.387
conv3_block4_out	0.768	0.780	k=6	0.952
conv4_block6_out	0.768	0.782	k=7,8	1.093
conv5_block3_out	0.699	0.727	k=3	1.470

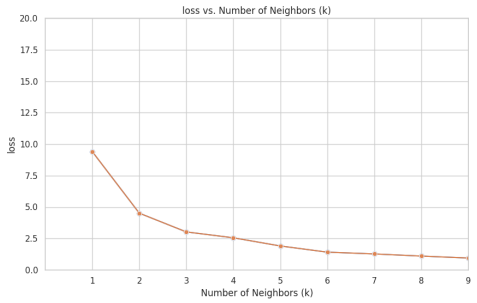
Tabela 4.3: Acurácia das camadas para o ajuste no contraste



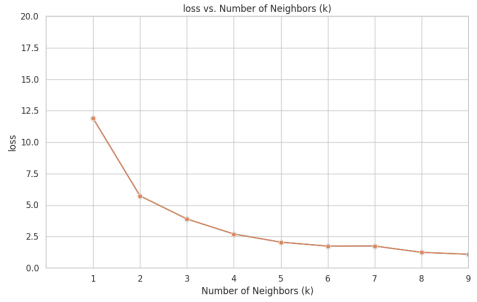
(a) *conv1_conv*



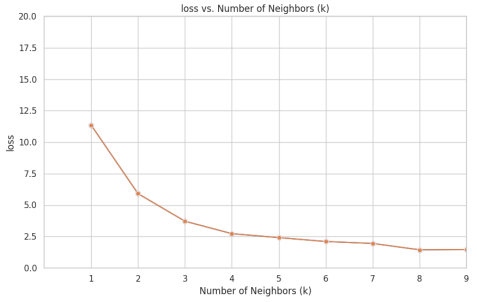
(b) *conv2_block3_out*



(c) *conv3_block4_out*



(d) *conv4_block6_out*



(e) *conv5_block3_out*

Figura 4.8: Gráficos de perda para o ajuste no contraste

Neste contexto, foi possível notar que em conjunto com valores de vizinhos, como este apresentado, será possível obter o melhor desempenho durante a identificação destas características.

A figura 4.9 permite visualizar a distribuição de classes para a alteração de contraste.

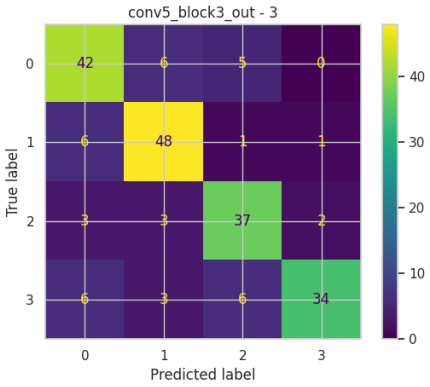


Figura 4.9: Matriz de confusão utilizando k=6

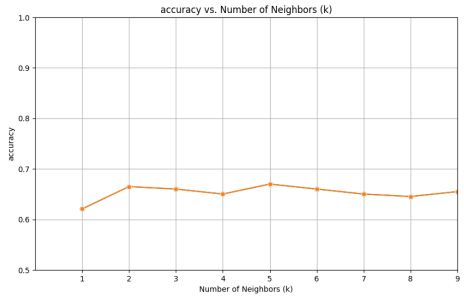
4.2.2 Aplicação de um limite no contraste

A análise utilizando a aplicação do CLAHE foi realizada juntamente com a aplicação de um valor limite para a equalização das imagens no dataset original. Essa abordagem foi adotada para melhorar o contraste das imagens adaptativamente, considerando as características locais de cada região da imagem.

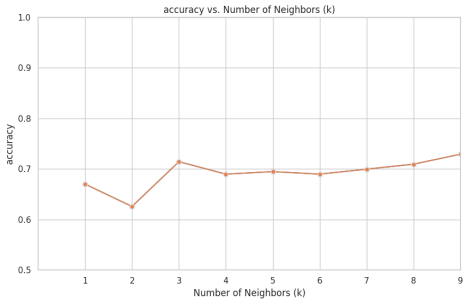
Além disso, também foram encontrados resultados significativos nas camadas mais profundas do modelo, especialmente nos blocos 4 e 5.

Em meio a figura 4.10 é possível notar que ao aplicar o ajuste de limite em meio ao contraste, houve uma melhora significativa, quando comparado ao ajuste 1.3, ao qual obteve uma acurácia de 76% e 1.093 para a perda, e quando realizado os testes em conjunto com o limite, se obteve um percentual de classificação de 83,7% e uma perda de 0.545. Todos os resultados podem ser visualizados em meio a tabela 4.4;

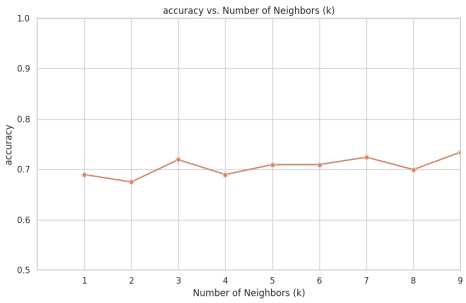
Um ponto interessante nos resultados, é a semelhança entre as camadas 4 e 5, que obtiveram resultados de acurácias diferentes, mas utilizando o mesmo valor de k. Neste



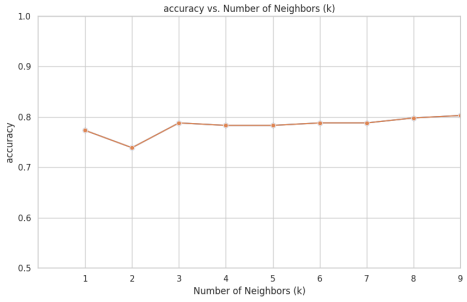
(a) conv1_conv



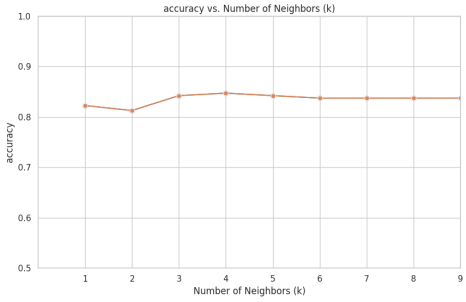
(b) *conv2_block3_out*



(c) *conv3_block4_out*

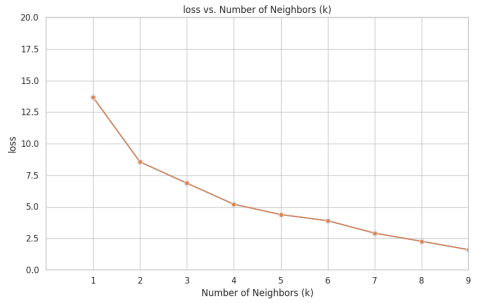


(d) *conv4_block6_out*

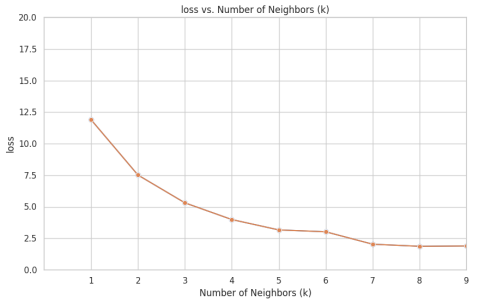


(e) *conv5_block3_out*

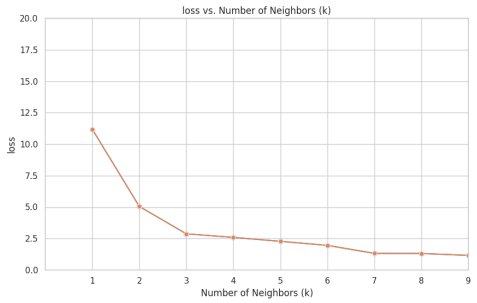
Figura 4.10: Gráficos de acurácia para ajuste com o CLAHE



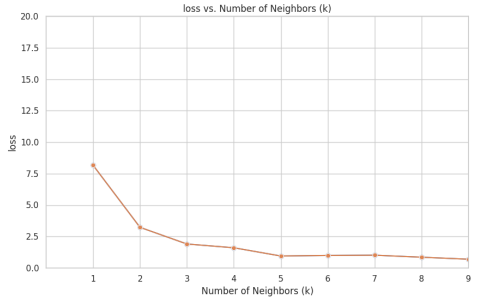
(a) *conv1_conv*



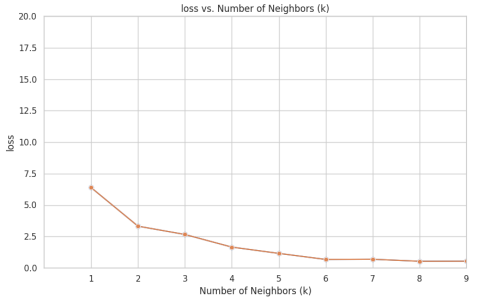
(b) *conv2_block3_out*



(c) *conv3_block4_out*



(d) *conv4_block6_out*



(e) *conv5_block3_out*

Figura 4.11: Gráficos de perda para ajuste com o CLAHE

contexto, é possível afirmar que a união entre um valor alto para k e as camadas mais profundas do modelo, permitiram uma classificação melhor, e também um percentual de perda mínimo.

Camada	Acurácia final	Maior acurácia	Valor k	Perda final
conv1_conv	0.650	0.660	k=9	1.604
conv2_block3_out	0.729	0.729	k=9	1.895
conv3_block4_out	0.733	0.733	k=9	1.164
conv4_block6_out	0.802	0.802	k=9	0.703
conv5_block3_out	0.837	0.831	k=4	0.545

Tabela 4.4: Acurácia das camadas para ajuste com o CLAHE

A figura 4.12 permite visualizar a distribuição de classes para a alteração de cor

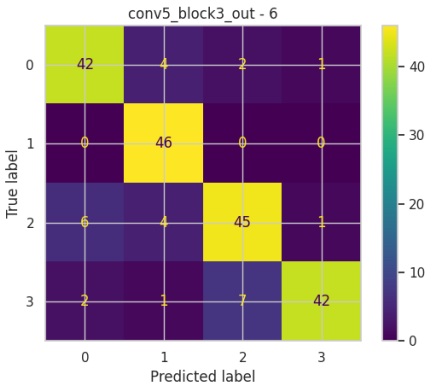


Figura 4.12: Matriz de confusão utilizando k=4

4.2.3 Técnicas de Albumentation

A análise das operações de rotação e reflexão horizontal (*Horizontal Flip*) permitiu observar a sensibilidade do modelo não apenas as mudanças nos pixeis das imagens, mas também a diferentes ângulos de apresentação. Utilizando a biblioteca Albumentations, foi possível aplicar essas transformações às imagens de forma eficaz.

Em meio aos efeitos desses ajustes nas imagens, notou-se a sensibilidade do modelo a diferentes ângulos de apresentação das imagens. Permitindo avaliar como as camadas do

modelo reagem a esses ajustes, fornecendo uma compreensão mais profunda sobre como o modelo processa e interpreta variações na orientação das imagens.

Rotação [-35,35]

Este ajuste foi realizado por meio da aplicação de uma rotação de -35 graus até +35 graus para cada imagem do dataset original. Conforme mostra a 4.13, é possível notar uma diferença dentre os demais testes, alcançando valores de acurácias bem diferentes. Neste contexto, notou-se que alterar a orientação de visualização da imagem, pode causar efeitos positivos sobre a classificação.

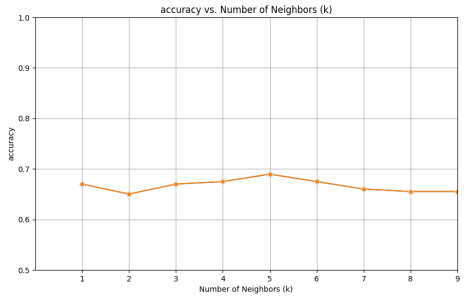
Para este ajuste, notou-se uma melhora em relação à acurácia, como mostra a figura 4.13e que obteve um percentual máximo de classificação de 85,2% para as camadas mais profunda, em meio ao uso de um valor de (k=3), no entanto, é possível notar que para valores maiores, como 6, também há um desempenho semelhante.

A tabela 4.5 permite comparar os valores de acurácia para este dataset, tendo como resultado duas camadas: 4 e 5. No entanto, se observado na figura 4.13 é possível notar que a 5 camada obteve o valor de acurácia maior, isso foi possível quando o valor de k era igual a 3. Para a 4 camada, o maior percentual atingido, foi de 0.772, em meio ao uso de um valor de (k=8).

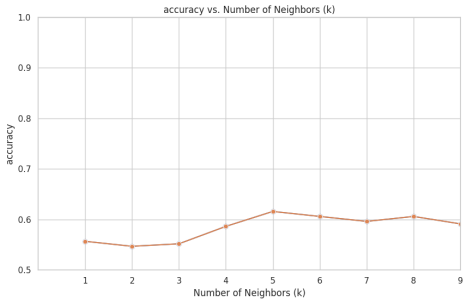
Em meio a figura de perda 4.14, também é possível notar uma semelhança, pois ambas as classificações, obtiveram uma variação mínima quando k atingiu um valor de vizinhos igual a 5.

Camada	**Acurácia final**	**Maior acurácia**	**Valor k**	**Perda final**
conv1_conv	0.650	0.673	k=5	1.960
conv2_block3_out	0.591	0.620	k=5,8	1.733
conv3_block4_out	0.684	0,711	k=7	1.216
conv4_block6_out	0.772	0.798	k=8	1.006
conv5_block3_out	0.852	0,877	k=3	0.533

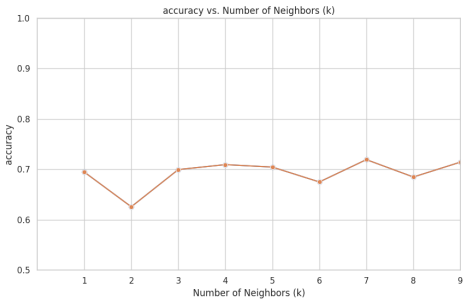
Tabela 4.5: Acurácia das camadas para ajuste da rotação em 35 graus



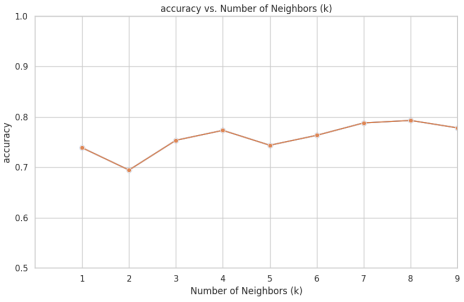
(a) conv1_conv



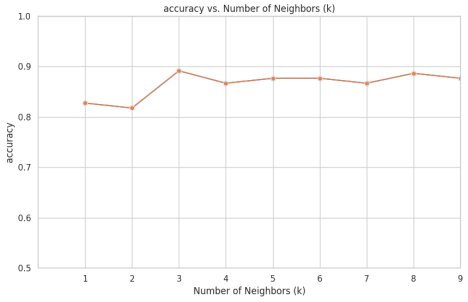
(b) *conv2_block3_out*



(c) *conv3_block4_out*

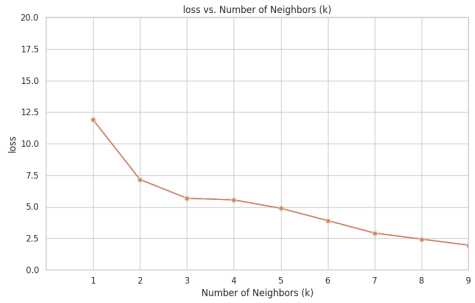


(d) *conv4_block6_out*

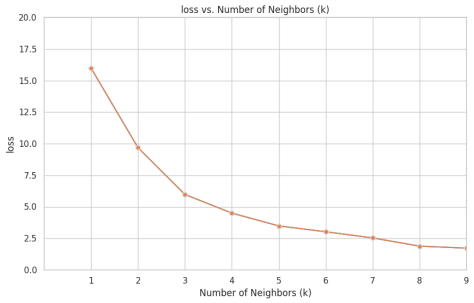


(e) *conv5_block3_out*

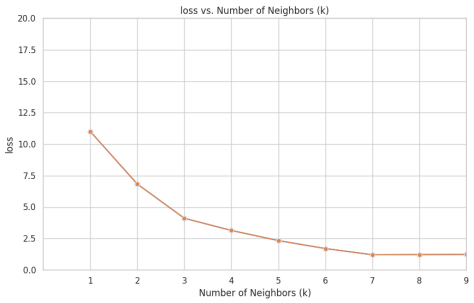
Figura 4.13: Gráficos de acurácia para ajuste da rotação em 35 graus



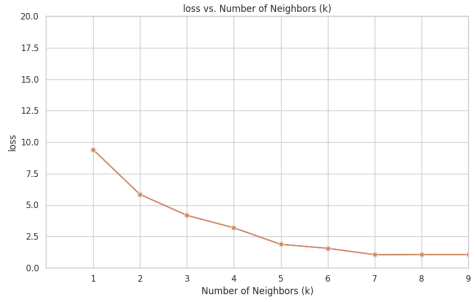
(a) *conv1_conv*



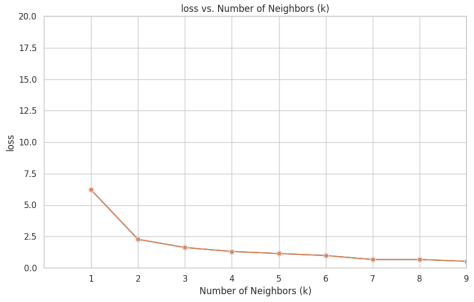
(b) *conv2_block3_out*



(c) *conv3_block4_out*



(d) *conv4_block6_out*

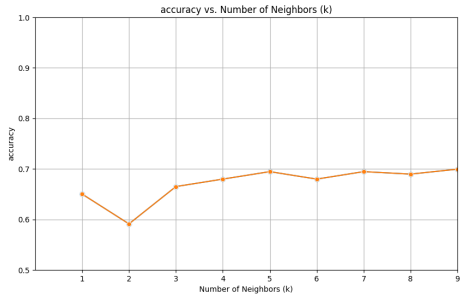


(e) *conv5_block3_out*

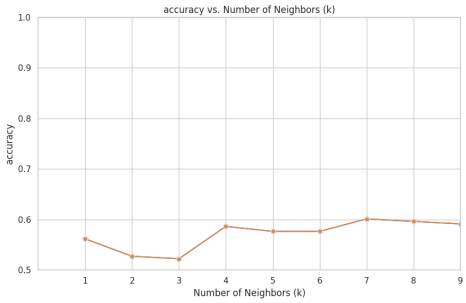
Figura 4.14: Gráficos de perda para ajuste da rotação em 35 graus

Rotação [-35,35] e *Flip* 0.2

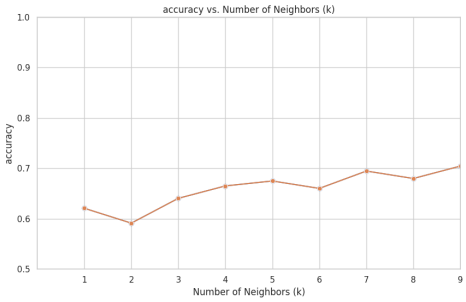
Após realizar apenas com um ajuste na rotação da imagem, foi aplicado de um valor para o *Horizontal Flip*, neste caso, foi aplicado uma quantidade de 0.2, sendo utilizado para espelhar a imagem de forma horizontal, semelhante a um espelho.



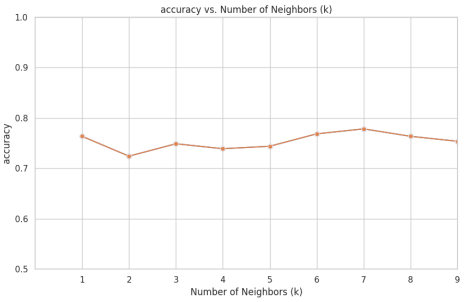
(a) conv1_conv



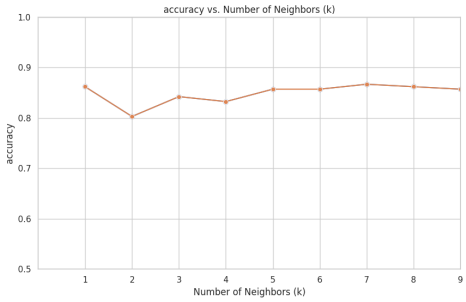
(b) *conv2_block3_out*



(c) *conv3_block4_out*



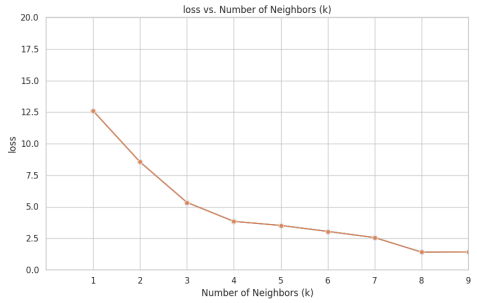
(d) *conv4_block6_out*



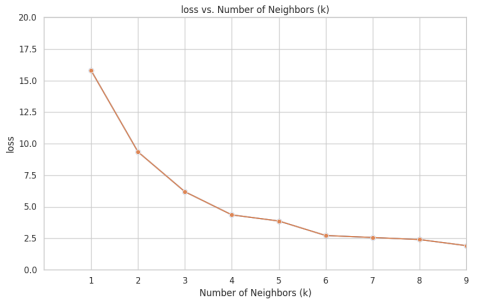
(e) *conv5_block3_out*

Figura 4.15: Gráficos de acurácia para ajuste da rotação de 35 graus e um flip de 0.2

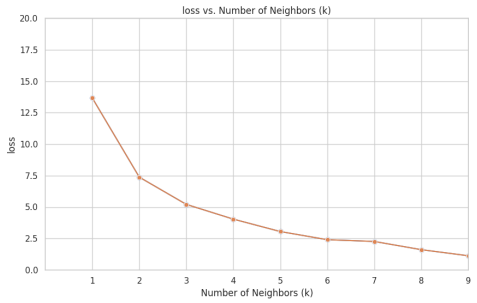
Essa alteração revelou resultados significativos para a análise, permitindo avaliar possíveis perspectivas sobre cada uma das camadas existentes nesta pesquisa. Neste contexto,



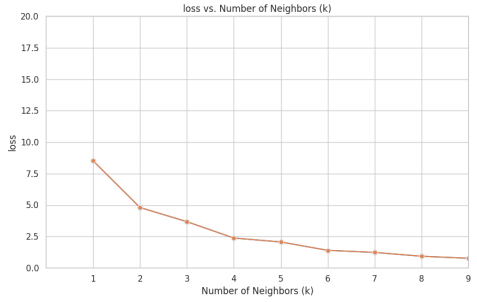
(a) *conv1_conv*



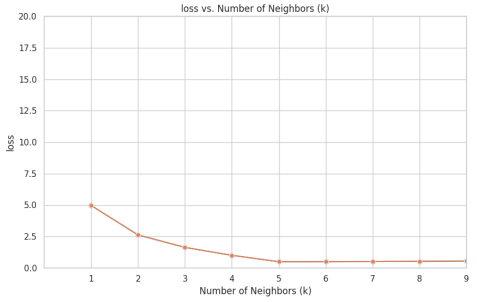
(b) *conv2_block3_out*



(c) *conv3_block4_out*



(d) *conv4_block6_out*



(e) *conv5_block3_out*

Figura 4.16: Gráficos de perda para ajuste da rotação de 35 graus e um flip de 0.2

observou-se uma discrepância entre as camadas mais profundas, onde as características presentes nos exemplos de treinamento conseguiram impactar mais a camada 5 do que a penúltima. Isso demonstra um impacto direto na complexidade do problema, uma vez que em vários conjuntos de imagens, é possível encontrar diferentes ângulos e orientações.

Camada	**Acurácia final**	**Maior acurácia**	**Valor k**	**Perda final**
conv1_conv	0.679	0,680	k=9	2.223
conv2_block3_out	0,577	0.600	k=7	2.559
conv3_block4_out	0.645	0.660	k=9	1.266
conv4_block6_out	0.758	0.818	k=7	0.764
conv5_block3_out	0.857	0,871	k=7	0.545

Tabela 4.6: Acurácia das camadas para ajuste da rotação de 35 graus e um flip de 0.2

A figura 4.16 mostra que a convergência para a menor taxa de erro, aconteceu mediante as camadas 4 e 5, já que após a iteração utilizando o valor de vizinhos (k) igual a 7, o erro se encontrava menor do que nas camadas iniciais (1, 2 e 3). Conforme a tabela 4.6 é possível visualizar as métricas coletadas ao final do processo de classificação.

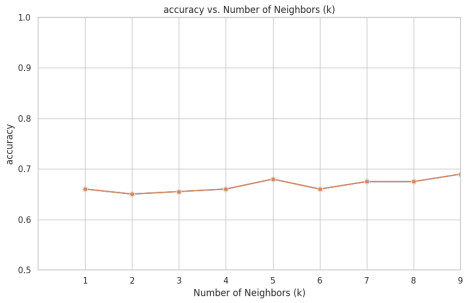
Rotação [-15,15] e *Flip* 0.2

Para este teste, foi implementado um valor de rotação menor, sendo de -15 graus até +15 graus e um *Flip* de 0.2. Assim como o primeiro teste utilizando a mudança de rotação, também foram notados resultados importantes, dentre eles um aumento sobre o valor de acurácia final, sendo este, a camada mais profunda (5 camada).

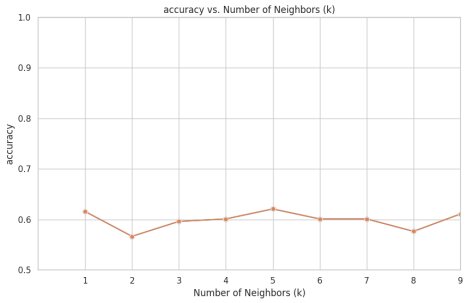
A figura 4.17 permite visualizar que em meio camada 3, à acurácia de 90.1% é atingida em dois momentos diferentes, sendo eles quando o número de vizinhos é igual a 6 e 9.

Em meio a figura 4.19d é possível visualizar uma crescente da acurácia, em relação à quantidade de vizinhos, desta forma, a camada 4 conseguiu obter um valor final de 85.2%, se mantendo na mesma faixa do que os testes realizados por meio da aplicação de 35 graus (negativo e positivo).

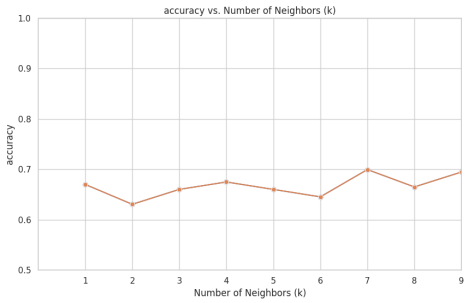
Por fim, a camada 5 ainda sim se manteve como a melhor dentre os testes, atingindo um percentual de 90,1% ao utilizar o valor de vizinhos (k) igual à 4.



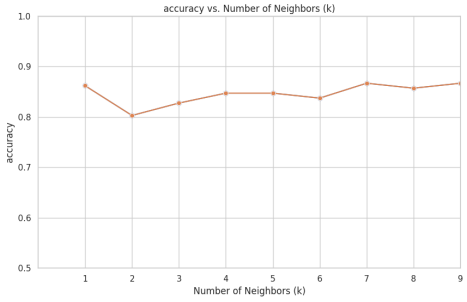
(a) conv1_conv



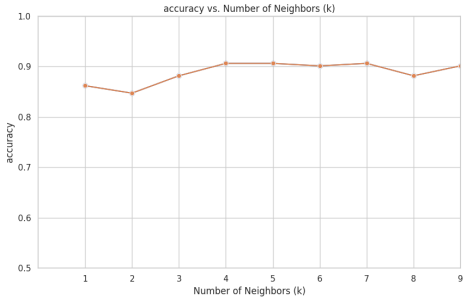
(b) *conv2_block3_out*



(c) *conv3_block4_out*

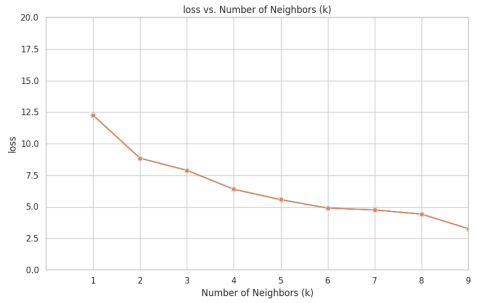


(d) *conv4_block6_out*

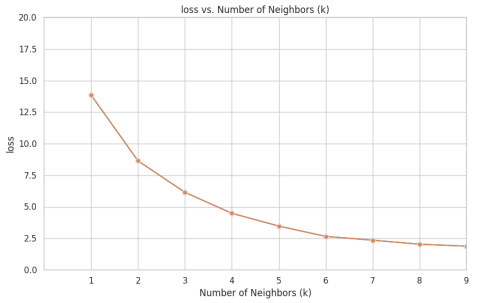


(e) *conv5_block3_out*

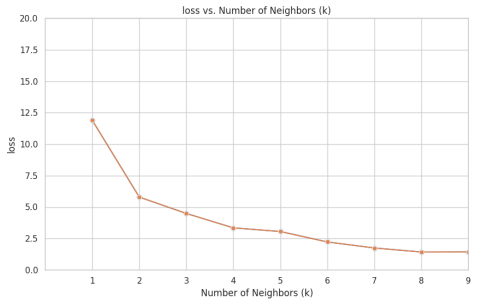
Figura 4.17: Gráficos de acurácia para ajuste da rotação de 15 graus e um flip de 0.2



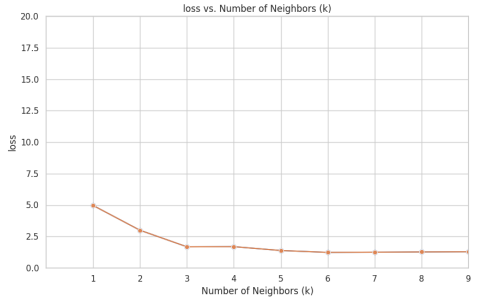
(a) *conv1_conv*



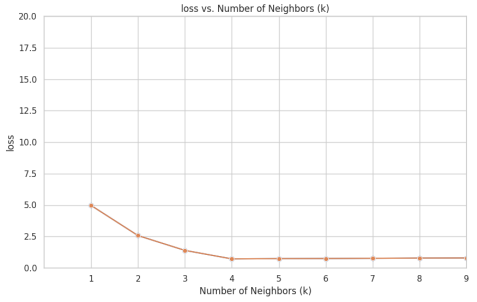
(b) *conv2_block3_out*



(c) *conv3_block4_out*



(d) *conv4_block6_out*



(e) *conv5_block3_out*

Figura 4.18: Gráficos de perda para ajuste da rotação de 15 graus e um flip de 0.2

A tabela 4.7 mostra que o percentual de perda alcançado, foi de 0.797, se mantendo acima do que os testes iniciais.

Camada	Acurácia final	Maior acurácia	Valor k	Perda final
conv1_conv	0.680	0,674	k=9	3.266
conv2_block3_out	0.615	0.615	k=5	1.855
conv3_block4_out	0.699	0.719	k=7	1.401
conv4_block6_out	0.862	0.852	k=9	1.212
conv5_block3_out	0.901	0.898	k=4	0.797

Tabela 4.7: Acurácia das camadas para ajuste da rotação de 15 graus e um flip de 0.2

Rotação [-15,15]

Diferentemente dos testes apresentados anteriormente, a aplicação de um valor de 15 graus apenas, possibilitou que o modelo alcançasse uma acurácia de 90% final, sendo que durante o processo de classificação, o mesmo consegue obter 90% de precisão em meio ao uso de um (k=5) para os vizinhos.

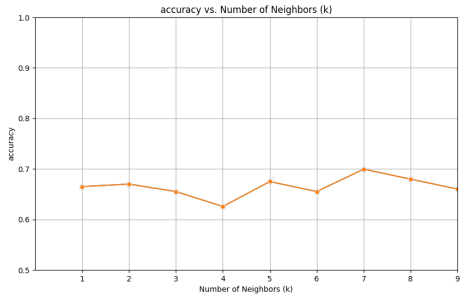
Esse percentual foi alcançado em meio a camada 5, como mencionado no início deste capítulo, possuí um bloco residual mais complexo, permitindo então que características mínimas possam ser identificadas.

Quando observado o gráfico de perda 4.20, nota-se uma instabilidade em termos de valores, sendo que após o aumento da quantidade de vizinhos, o valor de perda sofre uma variação mínima, se mantendo abaixo dos valores encontrados nos testes anteriores.

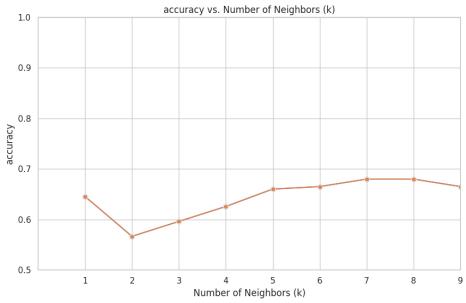
A tabela 4.8 permite visualizar às métricas descritas neste trabalho, bem como os valores de vizinhos com a melhor acurácia para este teste.

Camada	Acurácia final	Maior acurácia	Valor k	Perda final
conv1_conv	0.669	0.719	k=7	1.565
conv2_block3_out	0.665	0.690	k=7	2.478
conv3_block4_out	0.708	0.764	k=2	1.001
conv4_block6_out	0.842	0.879	k=6	1.084
conv5_block3_out	0.901	0.921	k=7	0.277

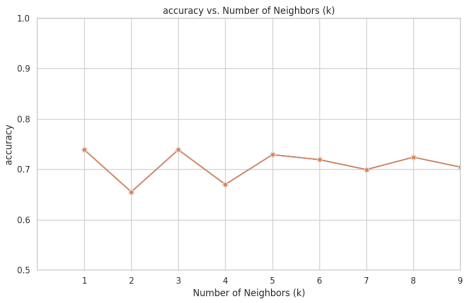
Tabela 4.8: Acurácia das camadas para ajuste da rotação de 15 graus



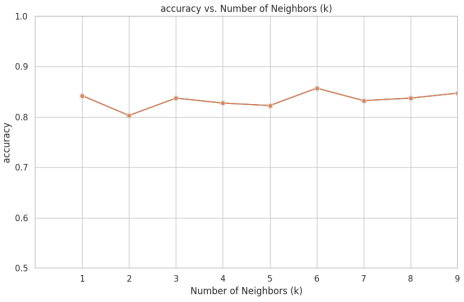
(a) conv1_conv



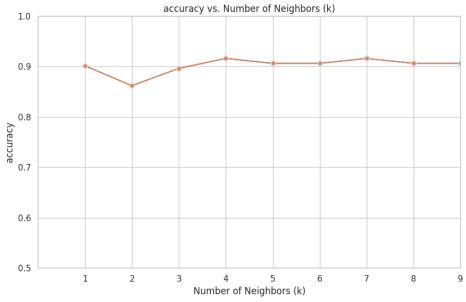
(b) *conv2_block3_out*



(c) *conv3_block4_out*

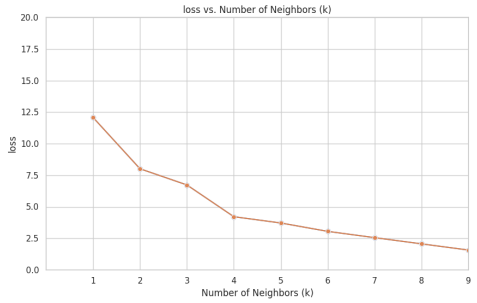


(d) *conv4_block6_out*

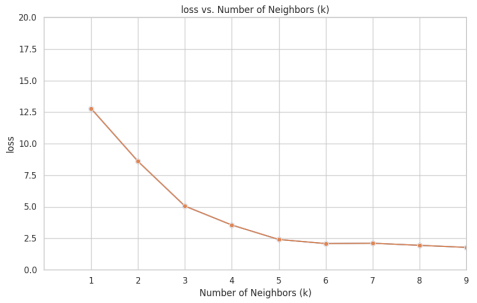


(e) *conv5_block3_out*

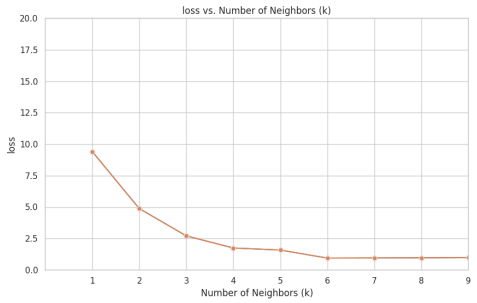
Figura 4.19: Gráficos de acurácia para ajuste da rotação de 15 graus



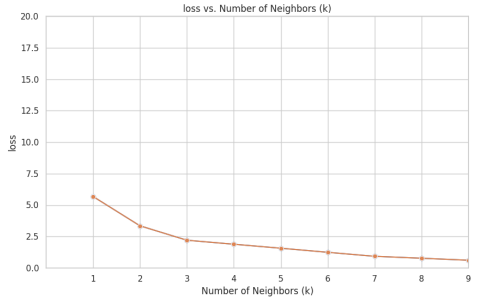
(a) *conv1_conv*



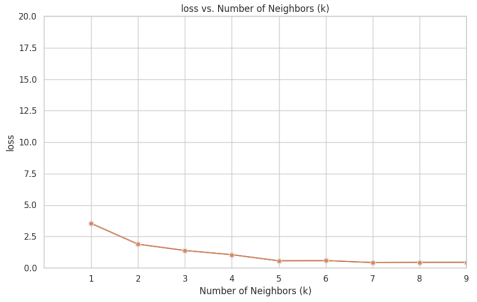
(b) *conv2_block3_out*



(c) *conv3_block4_out*



(d) *conv4_block6_out*



(e) *conv5_block3_out*

Figura 4.20: Gráficos de perda para ajuste da rotação de 15 graus

4.3 Resumo

Por meio dos dados obtidos, permitiu-se realizar uma análise completa sobre o comportamento do modelo de RNA em relação às técnicas aplicadas. Por exemplo, os ajustes na luminosidade possibilitaram uma adaptação mais eficaz das camadas superficiais. No entanto, ao aplicar um limite de contraste, como demonstrado pelo método CLAHE, observou-se uma mudança significativa, resultando em uma melhor adaptação das camadas mais profundas.

Além disso, os ajustes realizados por meio de técnicas de aumento de dados também demonstraram resultados satisfatórios para a pesquisa. Mesmo sem alterações diretas nas tonalidades das imagens, essas técnicas contribuíram para uma precisão de 90% do modelo. Isso ressalta a flexibilidade das técnicas de pré-processamento, que podem ser adaptadas conforme as necessidades específicas do problema em questão. Por exemplo, o ajuste da quantidade de vizinhos (k) permitiu a geração de uma ampla gama de métricas para análise detalhada. s

Capítulo 5

Conclusões e Trabalhos futuros

O treinamento realizado neste trabalho de dissertação utilizou o modelo ResNet50 pré-treinado com os pesos da ImageNet, explorando diferentes camadas de saída para cada bloco de convolução, visando avaliar o desempenho na tarefa de classificação. Foi utilizado um algoritmo de classificação KNN, variando o parâmetro k de 1 a 10, a fim de analisar o impacto na classificação.

A análise detalhada dos resultados permitiu extrair conclusões sobre o desempenho dos diversos algoritmos e sua adequação ao problema imposto. Por meio da visualização gráfica da taxa de acerto e erro do modelo em relação ao número de vizinhos para cada camada avaliada, conseguimos obter dados referentes aos impactos das modificações no conjunto de dados e nas camadas do modelo durante a classificação.

Conforme a pesquisa apresentada em Trabalhos relacionados, verificou-se que tanto imagens com variações de cores e texturas, podem impactar de forma diferente em meio a um modelo de rede neural. Neste contexto, verificou-se que a presença de cores em imagens, possuem mais impacto mediante as camadas iniciais, no entanto, imagens com maior concentração de textura, impactam em camadas mais profundas, podendo haver variações constantes entre as camadas.

Deste modo, foram utilizados diversas técnicas atuais do mercado, destacando-se no desenvolvimento de algoritmos para *Machine Learning* e *Deep Learning* e contribuindo significativamente para o avanço do conhecimento em meio a este campo.

Ao longo da análise realizada no Capítulo 4, foi possível observar as diferentes combinações que evidenciam a sensibilidade das camadas profundas dos modelos residuais, em especial a ResNet50, o que amplia a compreensão sobre a dinâmica desses modelos.

As técnicas e descobertas deste estudo não apenas aprimoram o entendimento teórico, mas também têm implicações práticas importantes. Podem ser aplicadas em uma variedade de outros contextos, facilitando tanto o desenvolvimento do conjunto de dados quanto o treinamento e a classificação dos modelos em diferentes domínios de aplicação.

Além disso, ressalta-se a importância contínua da pesquisa e da colaboração entre os pesquisadores da área. A abordagem adotada neste estudo mostra como a combinação de conhecimentos teóricos e práticos pode gerar avanços significativos e impactantes em meio a pesquisa de aprendizado de máquina.

Portanto, este trabalho não só contribui para o avanço do estado da arte em Machine Learning e Deep Learning, mas também serve como um ponto de partida para futuros trabalhos utilizando técnicas de pré-processamento para análise de imagens, melhoramento das características presentes no dataset, entre outras.

Este trabalho proporcionou uma série de resultados significativos que têm o potencial de impulsionar futuras pesquisas. Os dados obtidos em meio as análises não apenas facilitam a aplicação das análises realizadas nos modelos residuais com 50 camadas, mas também em outros tipos de redes neurais, permitindo também um estudo mais aprofundado sobre como as redes convolucionais (CNN) percebem cores e texturas, bem como a influência de diferentes parâmetros durante os estágios de treinamento e classificação. Essas descobertas não só aprimoram a compreensão sobre o funcionamento interno desses modelos, mas também fornecem uma visão mais ampla sobre o desenvolvimento de novas abordagens e técnicas no campo do aprendizado de máquina e visão computacional.

Bibliografia

[1] L. C. P. Travassos, «Inteligências Múltiplas,» *Revista de Biologia e Ciências da Terra*, vol. 1, n.º 2, p. 0, 2001.

[2] J. H. Flavell, M. H. S. Patto e J. Piaget, *A psicologia do desenvolvimento de Jean Piaget*. 1996.

[3] S. J. Russell e P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson, 2016.

[4] E. Brynjolfsson e A. McAfee, *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*. W. W. Norton & Company, 2017.

[5] N. Bostrom, *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, 2014.

[6] L. Floridi, «Soft Ethics and the Governance of the Digital,» *Philosophy & Technology*, vol. 32, n.º 1, pp. 1–8, 2019.

[7] J. McCarthy, M. L. Minsky, N. Rochester e C. E. Shannon, «A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence,» 1955.

[8] K. Mahesh, F. A. Khan e F. Mushtaq, «Machine Learning: An Overview,» *Materials Today: Proceedings*, vol. 21, pp. 1235–1239, 2020.

[9] B. Mahesh, «Machine learning algorithms-a review,» *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, n.º 1, pp. 381–386, 2020.

[10] I. Goodfellow, Y. Bengio e A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[11] J. Schmidhuber, «Deep learning in neural networks: An overview,» *Neural networks*, vol. 61, pp. 85–117, 2015.

[12] A. Rajkomar, E. Oren, K. Chen et al., «Scalable and accurate deep learning with electronic health records,» *NPJ Digital Medicine*, vol. 1, n.º 1, p. 18, 2018. DOI: 10.1038/s41746-018-0029-1.

[13] J. Kober, J. A. Bagnell e J. Peters, «Reinforcement learning in robotics: A survey,» *The International Journal of Robotics Research*, vol. 32, n.º 11, pp. 1238–1274, 2013. DOI: 10.1177/0278364913495721.

[14] P. Mbachu, P. Wu, H. Li, W. Li e Y. Zhang, «Machine learning applications in construction project management: A comprehensive review,» *Automation in Construction*, vol. 110, p. 102 951, 2020. DOI: 10.1016/j.autcon.2019.102951.

[15] C. S. FUJIKAWA, «Reconhecimento Facial utilizando Descritores de Textura e Aprendizado Não Supervisionado,» *Monografia (Bacharel em Ciências da Computação), UNESP (Universidade Estadual Paulista" Júlio de Mesquita Filho"), Rio Claro, Brazil*, 2016.

[16] C. H. C. Ribeiro, «A tutorial on reinforcement learning techniques,» em *Supervised Learning Track Tutorials of the 1999 International Joint Conference on Neuronal Networks*, 1999.

[17] D. A. G. Vieira, «Rede perceptron com camadas paralelas (PLP-Parallel Layer Perceptron),» 2006.

[18] S. Haykin, *Redes neurais: princípios e prática*. Bookman Editora, 2001.

[19] URL: https://cerebromente.org.br/n05/tecnologia/rna_i.htm.

[20] N. Gupta et al., «Artificial neural network,» *Network and Complex Systems*, vol. 3, n.º 1, pp. 24–28, 2013.

[21] L. Luo, «Architectures of neuronal circuits,» *Science*, vol. 373, n.º 6559, eabg7285, 2021.

[22] P. Kofuji e A. Araque, «G-protein-coupled receptors in astrocyte–neuron communication,» *Neuroscience*, vol. 456, pp. 71–84, 2021.

[23] Jan. de 2018. URL: https://www.deeplearningbook.com.br/o-neuronio-biologico-e-matematico/.

[24] *Capítulo 4 - o neurônio, biológico e matemático*, jan. de 2018. URL: https://www.deeplearningbook.com.br/o-neuronio-biologico-e-matematico/.

[25] C. A. da Silva Junior, M. R. Nanni, E. Cezar et al., «Rede neural artificial (perceptron) aliada a índices de vegetação na estimativa de áreas com plantas de soja,»

[26] J. C. Hay, B. E. Lynch e D. R. Smith, «Mark I perceptron operators' manual,» *Cornell Aeronautical Lab., Buffalo, NY, Rept. No. VG-1196-G-5*, 1960.

[27] Y. LeCun, «Deep Learning,» *Nature*, vol. 521, n.º 7553, pp. 436–444, 2015. DOI: 10.1038/nature14539.

[28] T. W. Rauber, «Redes neurais artificiais,» *Universidade Federal do Espírito Santo*, vol. 29, 2005.

[29] A. Krizhevsky, I. Sutskever e G. E. Hinton, «ImageNet Classification with Deep Convolutional Neural Networks,» *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[30] K. Simonyan e A. Zisserman, «Very Deep Convolutional Networks for Large-Scale Image Recognition,» *arXiv preprint arXiv:1409.1556*, 2014.

[31] I. Z. Mukti e D. Biswas, «Transfer Learning Based Plant Diseases Detection Using ResNet50,» em *2019 4th International Conference on Electrical Information and Communication Technology (EICT)*, 2019, pp. 1–6. DOI: 10.1109/EICT48899.2019.9068805.

[32] K. He, X. Zhang, S. Ren e J. Sun, «Deep Residual Learning for Image Recognition,» *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[33] J. Yosinski, J. Clune, Y. Bengio e H. Lipson, «How transferable are features in deep neural networks?» *Advances in Neural Information Processing Systems*, vol. 27, pp. 3320–3328, 2014.

[34] G. B. V. Junior, B. N. Lima, A. A. Pereira et al., «Métricas utilizadas para avaliar a eficiência de classificadores em algoritmos inteligentes,» *Revista CPAQV–Centro de Pesquisas Avançadas em Qualidade de Vida/ Vol*, vol. 14, n.º 2, p. 2, 2022.

[35] S. Rothe, «About the reliability of diagnostic statements: fundamentals about detection rates, false alarms, and technical requirements,» 2016.

[36] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, 2019.

[37] S. J. Pan e Q. Yang, «A survey on transfer learning,» *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, n.º 10, pp. 1345–1359, 2010. DOI: 10.1109/TKDE.2009.191.

[38] S. Levine, C. Finn e T. Darrell, «End-to-end training of deep visuomotor policies,» *Journal of Machine Learning Research*, vol. 17, n.º 39, pp. 1–40, 2016.

[39] K. Weiss, T. M. Khoshgoftaar e D. Wang, «A survey of transfer learning,» *Journal of Computer Science and Technology*, vol. 31, n.º 6, pp. 1213–1230, 2016. DOI: 10.1007/s11390-016-1676-y.

[40] Y. Bengio, A. Courville e P. Vincent, «Representation learning: A review and new perspectives,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, n.º 8, pp. 1798–1828, 2013. DOI: 10.1109/TPAMI.2013.50.

[41] J. Smith e M. Johnson, «A Survey of Computer Vision,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, n.º 7, pp. 145–167, 2013.

[42] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010.

[43] R. Gonzalez, R. Woods e S. Eddins, *Digital Image Processing*. Pearson/Prentice Hall, 2008.

[44] A. R. Smith, «Color gamut transform pairs,» *Computer graphics and image processing*, vol. 9, n.º 1, pp. 12–19, 1979.

[45] L. Chen e W. Wang, «Visual Perception Models: A Comprehensive Review,» *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, n.º 6, pp. 1546–1556, 2006.

[46] C. Poynton, *Digital Video and HD: Algorithms and Interfaces*. Elsevier, 2012.

[47] J. F. Hughes, A. van Dam, J. D. Foley e S. K. Feiner, «Computer graphics: principles and practice,» 2013.

[48] H. Cheng, X. Jiang, Y. Sun e J. Wang, «Color image segmentation: advances and prospects,» *Pattern Recognition*, vol. 34, n.º 12, pp. 2259–2281, 2001, ISSN: 0031-3203. DOI: https://doi.org/10.1016/S0031-3203(00)00149-7. URL: https://www.sciencedirect.com/science/article/pii/S0031320300001497.

[49] M. Sonka, V. Hlavac e R. Boyle, *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[50] R. M. Haralick e L. G. Shapiro, *Image analysis and computer vision*. SIAM, 1987.

[51] J. C. Rocha, «Cor luz, cor pigmento e os sistemas RGB e CMY,» *Revista Belas Artes*, vol. 3, n.º 2, 2010.

[52] E. Lotufo, «Cor e comunicação,» 2016.

[53] T. Mäenpää e M. Pietikäinen, «Classification with color and texture: jointly or separately?» *Pattern Recognition*, vol. 37, n.º 8, pp. 1629–1640, 2004, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2003.11.011. URL: https://www.sciencedirect.com/science/article/pii/S0031320303004321.

[54] A. Rosenfeld, *Digital picture processing*. Academic press, 1976.

[55] *Scalable Visual Attribute Extraction through Hidden Layers of a Residual ConvNet*.

[56] P. A. Rodrigues, «Classificação de doenças nas plantas utilizando Transfer Learning numa aplicação móvel,» tese de mestrado, Instituto Politecnico de Braganca (Portugal), 2019.

[57] G. Van Rossum e F. L. Drake Jr, *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995, vol. 620.

[58] M. Lutz e D. Ascher, *Aprendendo Python*, 2ª ed., Bookman, ed. dez. de 2007, ISBN: 9788577800131.

[59] URL: `https://www.tensorflow.org/about?hl=pt-br`.

[60] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. "O'Reilly Media, Inc.", set. de 2019, ISBN: 9781492032595.

[61] Keras, *Home - Keras Documentation*, 2019. URL: `https://keras.io/`.

[62] Scikit-learn, *scikit-learn: machine learning in Python — scikit-learn 0.20.3 documentation*, 2019. URL: `https://scikit-learn.org/stable/index.html`.

[63] URL: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler`.

[64] URL: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html#sklearn.preprocessing.LabelEncoder`.

[65] OpenCV, *About OpenCV*, 2018. URL: `https://opencv.org/about/`.