Universidade Estadual de Londrina

KAUEE ROCHA PUERTAS

DESENVOLVIMENTO DE UM ALGORITMO PARA RECONHECIMENTO DE OBJETOS COM REDES NEURAIS CONVOLUCIONAIS

LONDRINA
2024

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Puertas, Kauee Rocha.
Desenvolvimento de um Algoritmo para Reconhecimento de Objetos com Redes Neurais Convolucionais / Kauee Rocha Puertas. - Londrina, 2024.
51 f.

Orientador: Gilberto Fernandes Junior.
Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Graduação em Ciência da Computação, 2024.
Inclui bibliografia.

1. Reconhecimento de Objeto - TCC. 2. Rede Neural Convolucional - TCC. 3. Aprendizado de Máquina - TCC. 4. Inteligência Artificial - TCC. I. Junior, Gilberto Fernandes. II. Universidade Estadual de Londrina. Centro de Ciências Exatas. Graduação em Ciência da Computação. III. Título.

CDU 519

KAUEE ROCHA PUERTAS

DESENVOLVIMENTO DE UM ALGORITMO PARA RECONHECIMENTO DE OBJETOS COM REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof. Dr. Gilberto Fernandes Junior
Universidade Estadual de Londrina

Prof. Dr. Bruno Bogaz Zarpelão
Universidade Estadual de Londrina – UEL

Prof. Arthur Alexandre Artoni
Universidade Estadual de Londrina – UEL

Londrina, 24 de abril de 2024.

Este trabalho é dedicado às crianças adultas
que, quando pequenas, sonharam em se
tornar cientistas.

AGRADECIMENTOS

Primeiramente, gostaria de expressar minha gratidão à minha família por me apoiarem desde o início, sempre me incentivando a ir atrás do que eu realmente queria na minha vida e ajudando da maneira que podiam. Obrigado por estarem comigo até mesmo nos momentos mais complicados, isso foi por vocês.

À minha namorada, Gabriella, por sempre estar ao meu lado, inclusive nos momentos mais difíceis, quando sua presença e incentivo me dão forças para continuar.

Aos meus amigos, que estiveram ao meu lado, compartilhando as dificuldades e superando os desafios juntos.

A todos os professores que deram o melhor de si dentro da sala de aula e contribuíram de alguma forma para o meu crescimento, em especial ao meu orientador Gilberto pelo esforço para me ajudar e me acompanhar nessa caminhada.

Este trabalho não teria sido possível sem o apoio e o incentivo de cada uma dessas pessoas importantes em minha vida. A todos vocês, minha sincera gratidão.

"Um ser humano deve transformar informação em inteligência ou conhecimento. Tendemos a esquecer que nenhum computador jamais fará uma nova pergunta.
(Grace Hopper)

ROCHA, P. K. **Desenvolvimento de um Algoritmo para Reconhecimento de Objetos com Redes Neurais Convolucionais**. 2024. 51f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2024.

RESUMO

No campo da Inteligência Artificial, o reconhecimento de objetos é um dos desafios mais significativos e impactantes. A capacidade de identificar e classificar objetos em imagens impulsiona uma variedade de aplicações práticas, desde a visão computacional até a automação industrial, refletindo os avanços contínuos na interseção entre tecnologia e percepção humana. Nos últimos anos, tem havido um progresso notável na precisão e eficácia dos sistemas de reconhecimento de objetos, impulsionado pelo aprimoramento de algoritmos de aprendizado de máquina e pelo aumento da disponibilidade de conjuntos de dados vastos e diversificados. No entanto, o desenvolvimento e a otimização de algoritmos de reconhecimento de objetos utilizando CNNs apresentam desafios significativos, como a seleção adequada da arquitetura da rede, o treinamento eficiente com conjuntos de dados representativos, pré-processamentos, entre outros. Neste contexto, o presente trabalho explora o campo do reconhecimento de objetos, desenvolvendo um modelo para classificação de objetos em imagens por meio de uma abordagem baseada em redes neurais convolucionais e técnicas de aprendizado profundo. As etapas adotadas na concepção, implementação e avaliação do sistema proposto são discutidas, bem como uma análise dos resultados ao não obter uma boa acurácia e dificuldades encontradas ao longo do projeto. Ao final, são oferecidas considerações finais e sugestões para pesquisas futuras, visando aprimorar ainda mais os sistemas de reconhecimento de objetos e suas aplicações no mundo real.

Palavras-chave: Reconhecimento de Objeto. Inteligência Artificial. Rede Neural Convolucional. Aprendizado de Máquina.

ROCHA, P. K. **Development of an Algorithm for Object Recognition using Convolutional Neural Networks**. 2024. 51p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2024.

ABSTRACT

In the field of Artificial Intelligence, object recognition stands as one of the most significant and impactful challenges. The ability to identify and classify objects in images drives a variety of practical applications, from computer vision to industrial automation, reflecting ongoing advancements at the intersection of technology and human perception. In recent years, there has been notable progress in the accuracy and effectiveness of object recognition systems, propelled by improvements in machine learning algorithms and the increasing availability of vast and diverse datasets. However, the development and optimization of object recognition algorithms using CNNs pose significant challenges, such as selecting appropriate network architectures, efficient training with representative datasets, preprocessing, among others. In this context, the present work explores the field of object recognition by developing a model for classifying objects in images through a convolutional neural network-based approach and deep learning techniques. The steps taken in the design, implementation, and evaluation of the proposed system are discussed, along with an analysis of the results and challenges encountered throughout the project. Finally, concluding remarks and suggestions for future research are provided, aiming to further enhance object recognition systems and their real-world applications.

Keywords: Object Recognition. Artificial Intelligence. Convolutional Neural Networks. Machine Learning.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AM - Aprendizado de Máquina

BBOX - *Bounding Box*

CG - Computação Gráfica

CNN - Rede Neural Convolucional

COCO - *Common Objects in COntext*

DA - *Data Augmentation*

HOG - Histograma de Orientações dos Gradientes

IA - Inteligência Artificial

PDI - Processamento de Imagem

RELU - *Rectified Linear Unit*

SVM - Máquinas de Vetor de Suporte

SUMÁRIO

1 INTRODUÇÃO

No campo da Inteligência Artificial, o reconhecimento de objetos representa um dos desafios mais significativos e impactantes. A capacidade de identificar e classificar objetos em imagens não apenas impulsiona uma variedade de aplicações práticas, desde a visão computacional até a automação industrial, mas também reflete os avanços contínuos na interseção entre tecnologia e percepção humana. Nos últimos anos, testemunhamos um progresso impressionante na precisão e na eficácia dos sistemas de reconhecimento de objetos, impulsionado pelo aprimoramento de algoritmos de aprendizado de máquina e pelo aumento da disponibilidade de conjuntos de dados vastos e diversificados [5] [6].

Esses avanços têm sido especialmente notáveis na área de reconhecimento de imagens, onde algoritmos complexos e modelos de rede neural convolucional (CNN) têm sido desenvolvidos para identificar objetos em fotografias, vídeos e fluxos de câmeras em tempo real [7] [8]. Com a capacidade de diferenciar entre uma ampla gama de classes de objetos com uma precisão cada vez maior, esses sistemas estão revolucionando setores que vão desde a medicina e a agricultura até a segurança e o entretenimento [9].

No entanto, o desenvolvimento e a otimização de algoritmos de reconhecimento de objetos utilizando CNNs apresentam desafios significativos. Estes incluem a seleção adequada da arquitetura da rede, o treinamento eficiente com conjuntos de dados representativos, a regularização para evitar o *overfitting*, e a interpretação e análise dos resultados obtidos [10]. Além disso, a implementação deve considerar requisitos de desempenho em tempo real, eficiência computacional e escalabilidade para lidar com grandes volumes de dados.

Neste contexto, o presente trabalho busca explorar o campo do reconhecimento de objetos, desenvolvendo um modelo para classificação de objetos em imagens por meio de uma abordagem baseada em redes neurais convolucionais e técnicas de aprendizado profundo. Serão discutidos diferentes aspectos relacionados à concepção e treinamento do modelo, bem como uma análise dos resultados e dificuldades encontradas ao longo do projeto.

A estrutura deste trabalho é organizada da seguinte maneira: no Capítulo 2, é apresentada a fundamentação teórica, explorando os conceitos essenciais para este estudo. Já no Capítulo 3, são descritos os métodos e etapas adotados para conceber, implementar e avaliar o sistema proposto. No Capítulo 4, os resultados são apresentados e analisados em detalhes. Finalmente, o Capítulo 5 traz as considerações finais deste trabalho e sugestões para pesquisas futuras.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Processamento de Imagem (PDI)

O processamento de imagem se derivou do processamento de sinais, onde os sinais, assim como as imagens, carregam consigo uma informação. Processar essa imagem se resume a transformá-la repetidamente com o propósito de extrair mais facilmente a informação presente nela [11].

A principal funcionalidade do processamento digital de imagens reside em disponibilizar instrumentos que simplifiquem a detecção e extração de dados presentes na imagem, com vistas à subsequente interpretação. Para o fim da produção de outras imagens já contendo informações específicas, extraídas e realçadas, sistemas dedicados de computação são empregados na análise e manipulação interativa das imagens originais [12].

O processamento de imagens digitais abrange uma ampla escala de hardware, software e fundamentos teóricos. As etapas definidas a seguir, e ilustradas na Figura 1, representam o fluxo geral de processamento de imagem digital e são fundamentais em várias aplicações, incluindo reconhecimento de padrões, visão computacional, medicina (diagnóstico por imagem), automação industrial e muitas outras [13].

1. Aquisição de Imagem Digital: Captura da imagem em formato digital.

2. Pré-processamento: Melhoramento da qualidade da imagem para aumentar as chances de sucesso nas etapas subsequentes. Isso envolve técnicas como realce de contraste e remoção de ruído.

3. Segmentação: Divisão da imagem de entrada em partes distintas. Geralmente, essa etapa é uma das mais desafiadoras no Processamento de Imagem Digital (PDI). Por exemplo, no reconhecimento de caracteres, a segmentação desempenha um papel fundamental ao extrair caracteres individuais e palavras do fundo da imagem.

4. Representação: Conversão dos dados da imagem para um formato adequado ao processamento computacional. Isso inclui decidir se os dados devem ser representados como fronteiras ou como regiões completas. Fronteiras capturam apenas as linhas que delimitam os objetos, focando nos contornos, o que é útil para detecção de bordas e reconhecimento de formas. Regiões completas capturam toda a área interna dos objetos, incluindo bordas e conteúdo, sendo úteis para análise de textura e segmentação detalhada.

5. Extração ou Seleção de Características: Identificação e extração de características que contenham informações quantitativas de interesse ou que sejam essenciais para a discriminação entre diferentes classes de objetos. No reconhecimento de caracteres, por exemplo, buracos e concavidades são características cruciais que ajudam a diferenciar partes do alfabeto.

6. Reconhecimento: Atribuição de rótulos a objetos com base nas informações fornecidas pelas características extraídas.

7. Interpretação: Atribuição de significado a um conjunto de objetos reconhecidos, contextualizando-os em uma perspectiva mais ampla.

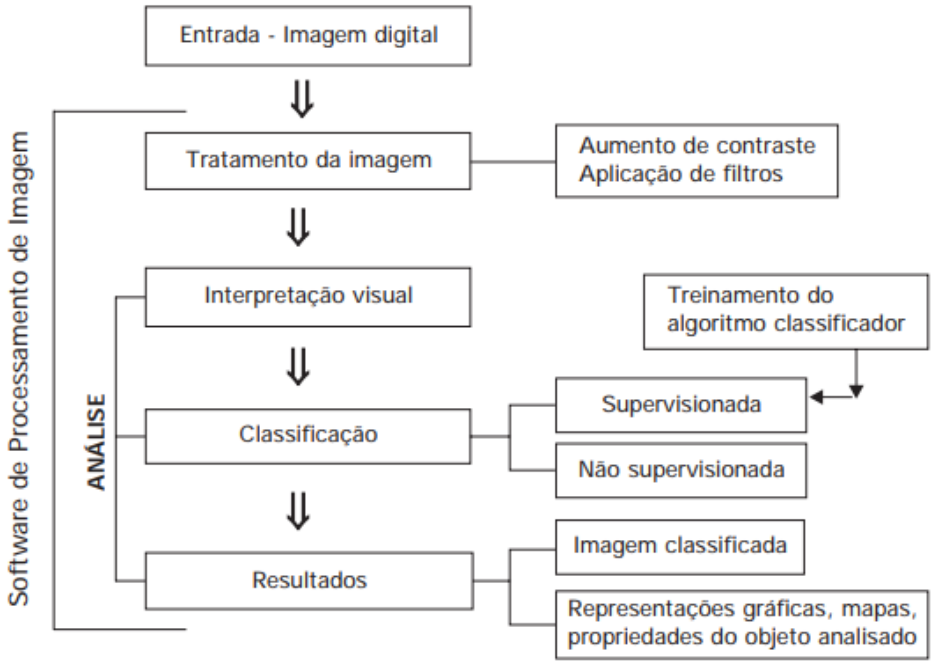

Figura 1 – Alguns dos passos do processamento representados. [1]

2.1.1 Relação entre Processamento Digital de Imagens e Computação Gráfica

A área de Processamento Digital de Imagens (PDI) e de Computação Gráfica (CG) tem crescido significativamente devido à sua ampla aplicabilidade nos dias de hoje. Em termos comparativos, essas áreas, embora relacionadas, são distintas. A CG busca criar imagens foto-realísticas de cenas tridimensionais, que são representações visuais com largura, altura e profundidade, geradas por computador. Já o PDI visa reconstruir uma cena tridimensional a partir de uma imagem real capturada por uma câmera [14].

A Computação Gráfica parte de dados precisos para gerar uma imagem, focando na síntese de cenas tridimensionais. Por outro lado, o PDI parte de uma imagem exis-

tente, geralmente capturada por uma câmera, e busca extrair e reconstruir a informação tridimensional a partir dela, realizando um processo de análise [11].

O cenário que engloba todas as disciplinas que incorporam elementos de processamento de informações visuais, com especial ênfase em áreas como CG e PDI, é definido por alguns autores como Computação Visual [14].

2.2 Aprendizado de Máquina

Aprendizado de Máquina (AM) é uma categoria da Inteligência Artificial (IA) que visa capacitar computadores a fazer previsões utilizando conjuntos de dados e algoritmos. Sua característica mais relevante é capacitar sistemas computacionais a aprender e melhorar seu desempenho de maneira autônoma, ao invés de dependerem apenas de programação explícita [15].

2.2.1 Tipos de Aprendizado

O treinamento de modelos de Aprendizado de Máquina pode ser feito de várias maneiras, dependendo dos dados disponíveis e do problema em questão [16]. Alguns dos tipos de aprendizado mais populares são:

- Aprendizado supervisionado: neste cenário, os conjuntos de dados utilizados no treinamento são rotulados, o que permite que o modelo verifique as respostas corretas para medir sua precisão. Para problemas de classificação e regressão, essa técnica é particularmente útil porque é imprescindível prever um resultado específico com base em entradas conhecidas [17].

- Aprendizado não supervisionado: aqui, o modelo de machine learning recebe apenas dados não rotulados. Não haver rótulos torna a avaliação do desempenho do modelo mais difícil. Mas quando o objetivo é encontrar padrões ou agrupar dados naturalmente, essa técnica é útil para problemas de segmentação [17].

- Aprendizado semi-supervisionado: apenas uma parte dos dados utilizados no treinamento é rotulada nesse contexto. Isso é vantajoso em situações onde rotular dados manualmente é custoso ou demorado, pois permite aproveitar ao máximo os dados rotulados disponíveis, combinando-os com dados não rotulados para melhorar o desempenho do modelo [17].

- Aprendizado por reforço: o modelo aprende por meio de feedbacks recebidos a partir de suas ações. É comumente utilizado em problemas de tomada de decisão sequencial, onde o agente deve aprender a interagir com um ambiente dinâmico para maximizar uma recompensa cumulativa ao longo do tempo [18].

2.3 Redes Neurais Artificiais

De uma forma mais geral, uma rede neural é um sistema desenvolvido para imitar o funcionamento do cérebro em uma tarefa específica, sendo implementada por meio de componentes eletrônicos ou simulada por propagação em um computador digital. Para obterem um bom desempenho, as redes neurais utilizam uma extensa interconexão de unidades computacionais simples, frequentemente referidas como "neurônios" ou unidades de processamento [19].

Já uma rede neural artificial (RNA) é um modelo computacional baseado na estrutura do cérebro humano, onde é composta por elementos de processamento simples chamados neurônios artificiais, que aplicam funções matemáticas e funções de ativação aos dados, produzindo uma resposta distinta. Estes neurônios são agrupados em camadas e conectados uns aos outros por pesos ajustáveis. Para a extração de dados e o armazenamento de conhecimento na rede, é necessário o processo de ajuste desses pesos, conhecido como treinamento ou aprendizado. A generalização, ou a capacidade de fornecer respostas precisas para dados não vistos durante o treinamento, é um uso significativo das redes neurais [20].

As redes neurais artificiais são frequentemente utilizadas para resolver desafios complexos, especialmente nos casos em que o comportamento das variáveis é difícil de determinar com precisão. A habilidade de aprender com exemplos e extrapolar o que sabem para criar modelos não-lineares é uma característica fundamental das redes neurais. Devido a essa capacidade, elas são excepcionalmente eficazes em vários contextos, incluindo a análise espacial, um ramo da ciência geoespacial que desempenha um papel importante na compreensão e na modelagem de fenômenos que ocorrem em um contexto geográfico. Esse campo se concentra na exploração e na interpretação de dados que possuem uma componente espacial, como coordenadas geográficas, para identificar padrões, relações e tendências influenciadas pela localização. Ao utilizar técnicas avançadas, como a modelagem de dados geográficos por meio de redes neurais artificiais, podemos não apenas analisar a distribuição e a interação de variáveis ambientais complexas, mas também prever e compreender melhor o comportamento desses fenômenos em diferentes cenários [21].

A topologia de uma rede neural é definida por várias variáveis importantes, conforme mencionado por Santos [22]. Isso inclui:

1. O número de nós na camada de entrada: esta variável determina quantas variáveis serão utilizadas para alimentar a rede neural. Normalmente, essas variáveis são escolhidas com base na sua importância para o problema em estudo.

2. O número de camadas ocultas e o número de neurônios em cada camada: as camadas

ocultas são responsáveis por processar e transformar as entradas da rede antes de chegar à camada de saída. O número de camadas ocultas e o número de neurônios em cada uma dessas camadas são decisões importantes na arquitetura da rede neural e podem afetar significativamente o seu desempenho.

3. O número de neurônios na camada de saída: esta variável determina quantos neurônios estarão na camada de saída da rede neural. O número de neurônios na camada de saída depende do tipo de problema que está sendo resolvido e do formato da saída desejada (por exemplo, classificação, regressão, etc.).

Ao definir essas variáveis, é possível configurar uma rede neural com uma topologia adequada para resolver o problema em questão de forma eficaz. Na Figura 2, temos uma representação gráfica de uma rede neural artificial. A rede tem duas entradas, duas camadas ocultas compostas por cinco neurônios cada, e duas saídas.

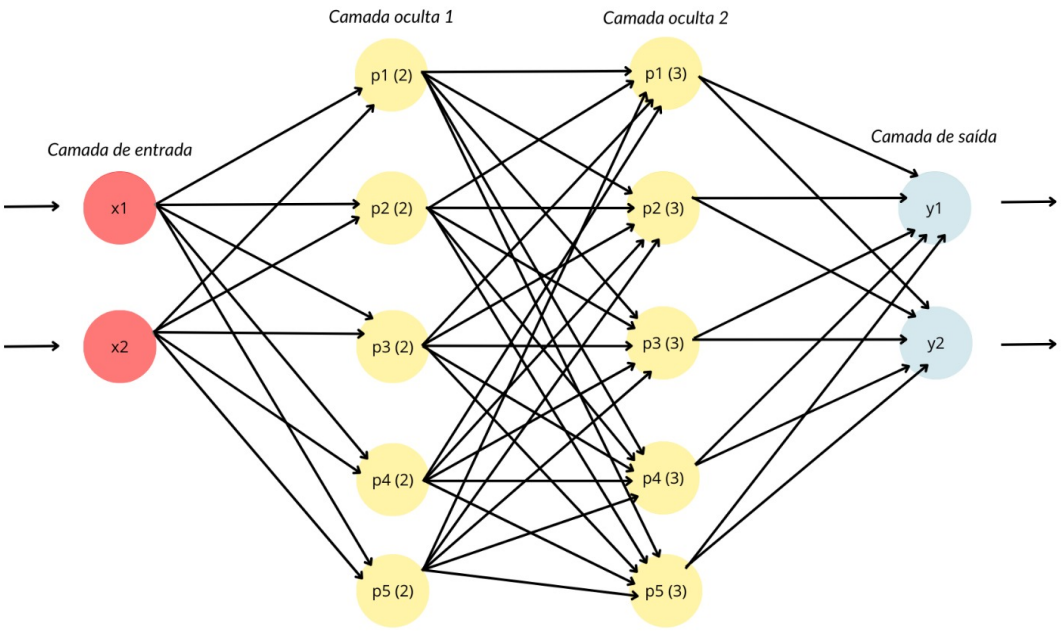

Figura 2 – Exemplo gráfico de uma rede neural artificial. Elaboração: Autor

Existem várias metodologias sugeridas na literatura, como Hirose et al. [23], Arai [24] e Fujita [25], que devem ser levadas em consideração ao calcular o número de neurônios na camada oculta. Essa decisão não é apenas empírica, mas também deve ser considerada a possibilidade de *overfitting*, ou seja, a memorização excessiva de dados de treinamento, como apontado no trabalho de Silva [26]. Por outro lado, *underfitting*, que ocorre quando o modelo não consegue capturar corretamente a complexidade dos dados, podendo ser resultado de um número insuficiente de neurônios.

2.3.1 Funções de Ativação

2.3.1.1 ReLu

A função de ativação ReLU (Rectified Linear Unit) é amplamente reconhecida por sua vantagem computacional de convergir rapidamente durante o treinamento de redes neurais, conforme Gharat [27]. Esta função de ativação é expressa como

$$f(x) = \max(0, x) \tag{2.1}$$

Onde os valores negativos são substituídos por zero e valores positivios permanecem inalterados, como ilustrado na Figura 3:

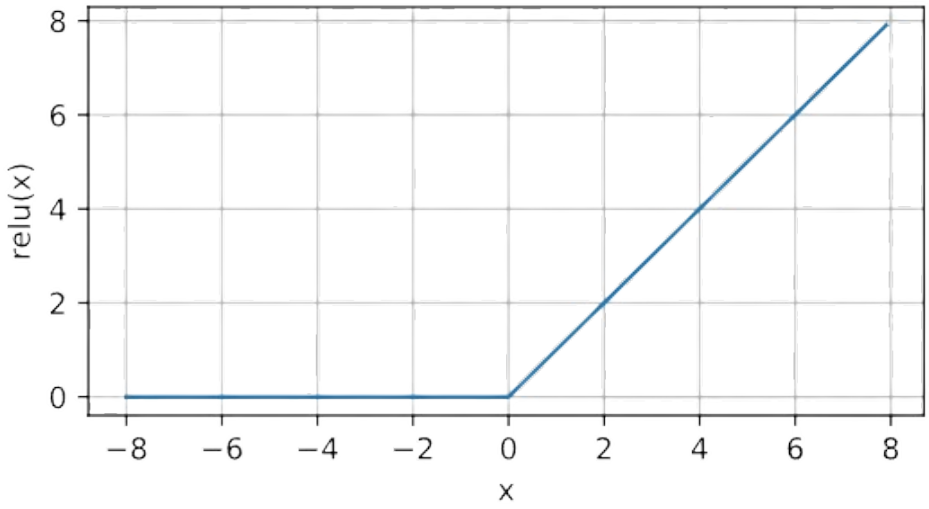

Figura 3 – Gráfico da função ReLu [2].

A principal vantagem da ReLU reside na sua eficiência computacional, pois a sua operação é simples e não exige cálculos exponenciais. Além disso, a função ReLU ajuda a mitigar o problema do desvanecimento do gradiente, comum em redes neurais profundas, ao permitir que gradientes positivos fluam livremente durante a retropropagação.

O desvanecimento do gradiente é um problema que ocorre durante o treinamento de redes neurais profundas, onde os gradientes das camadas mais profundas se tornam extremamente pequenos à medida que são propagados para trás através da rede durante o processo de retropropagação. Isso acontece especialmente em redes profundas com funções de ativação que possuem derivadas muito pequenas ou muito grandes, como as funções logísticas em redes antigas. Quando os gradientes se tornam muito pequenos, as atualizações dos pesos se tornam insignificantes e a rede neural tem dificuldade em aprender, o que leva a um treinamento lento ou estagnado. Portanto, a ReLU permite que a informação relevante seja transmitida e atualizações significativas nos pesos sejam feitas durante o treinamento, contribuindo para um treinamento mais eficiente e estável de redes neurais profundas [28].

Embora a ReLU tenha sido amplamente adotada devido à sua eficácia computacional e ao seu papel na aceleração da convergência das redes neurais, é importante observar que ela pode apresentar o problema conhecido como "dying ReLU", onde alguns neurônios podem se tornar inativos durante o treinamento e nunca mais se ativarem novamente [29]. Essa questão motivou o desenvolvimento de variações da ReLU, como a Leaky ReLU e a Parametric ReLU, que visam mitigar esse problema [30] [31].

Assim, a função ReLU desempenha um papel significativo no campo das redes neurais, oferecendo uma combinação única de eficiência computacional e capacidade de acelerar o processo de treinamento, enquanto também inspira o desenvolvimento de variantes para lidar com suas limitações.

2.3.1.2 Softmax

Já a função *softmax*, como discutida por Barber, Saad e Sollich [32], é uma função de ativação comumente utilizada em problemas de classificação. Quando se trata de conjuntos de dados com várias classes, ela é especialmente adequada, o que para o caso do desenvolvimento do reconhecedor de objeto que envolveu vários níveis de classificação desse trabalho é especialmente verdadeiro.

Sua principal característica é sua capacidade de transformar um vetor de números reais em um vetor de probabilidades, onde cada elemento do vetor de saída representa a probabilidade de pertencer a uma determinada classe. Isso é especialmente útil em problemas de classificação, pois fornece uma interpretação direta das previsões do modelo em termos de probabilidade de pertencer a cada classe.

A função *softmax* é calculada aplicando-se a exponencial a cada elemento do vetor de entrada e, em seguida, normalizando os resultados para que a soma de todas as probabilidades seja igual a 1. Isso garante que as saídas da função *softmax* estejam na faixa de 0 a 1 e que a soma de todas as probabilidades seja igual a 1, tornando-as interpretáveis como probabilidades.

2.3.2 Rede Neural Convolucional (CNN)

As redes neurais convolucionais (CNNs) são arquiteturas de destaque no campo das redes neurais artificiais. Elas se destacam por incorporar a operação de convolução em uma ou mais camadas, permitindo a extração eficiente de características em dados de entrada. A necessidade premente de treinar redes neurais de forma eficiente impulsionou o desenvolvimento e a adoção generalizada dessas arquiteturas, evidenciando sua importância e relevância na área da aprendizagem profunda [33].

As CNNs são frequentemente empregadas em cenários onde há uma grande quantidade de dados rotulados disponíveis. Elas apresentam diversas vantagens, tais como a

capacidade de extrair características relevantes por meio do aprendizado de transformações e a dependência de um número menor de parâmetros ajustáveis em comparação com redes totalmente conectadas com o mesmo número de camadas ocultas.

Além disso, as CNNs são amplamente reconhecidas por sua aplicação bem-sucedida em tarefas de processamento de imagens, devido à sua capacidade de extrair características das imagens por meio das operações de convolução e agrupamento (*pooling*). Ao contrário das redes neurais convencionais, que tendem a difundir as informações espaciais entre as camadas, as CNNs preservam a organização espacial dos pixels de entrada. Isso ocorre porque elas processam a entrada por meio de camadas convolucionais ou camadas de *pooling*, mantendo a localidade relativa dos pixels enquanto representam suas informações de maneira diferente.

2.3.2.1 Convolução

A operação de convolução consiste em aplicar um filtro sobre um sinal de entrada, resultando em um conjunto de características derivadas desse sinal. Essa operação pode ser interpretada como uma função que atribui pesos distintos para diferentes regiões da entrada. Quando utilizada em conjunto com redes neurais, as operações de convolução estão geralmente associadas ao processamento de imagens, onde são amplamente empregadas para extrair características relevantes.

2.3.2.2 Camada Convolucional

Nas redes neurais convolucionais (CNNs), as camadas convolucionais desempenham um papel essencial, aplicando operações de convolução sobre suas entradas. Essa operação consiste em aplicar um filtro sobre toda a extensão da entrada para extrair informações relevantes. Cada iteração do filtro cobre diferentes regiões da entrada, percorrendo toda a imagem para realizar a convolução. A Figura [4] ilustra a operação com uma entrada RGB.

Os filtros em uma camada convolucional, também chamados de kernels, definem como a filtragem é realizada. Eles consistem em uma matriz de pesos de tamanho fixo, que é deslocada sobre segmentos da imagem de entrada para gerar um mapa de características. Durante o treinamento da camada convolucional, os parâmetros ajustáveis incluem os pesos dos filtros e um valor de viés para cada filtro.

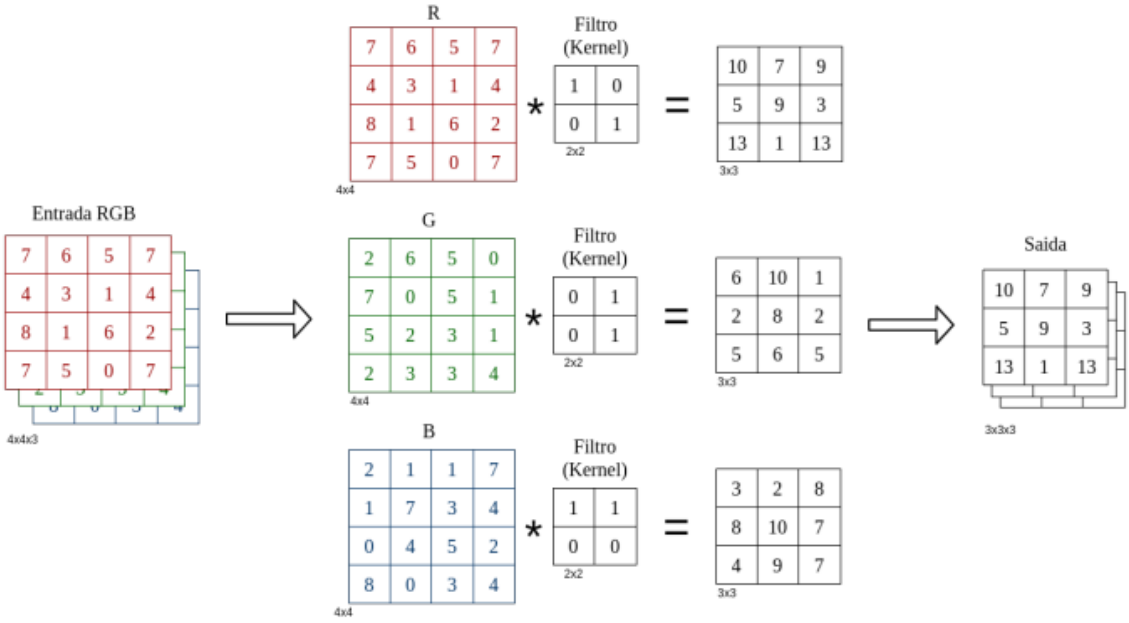

Figura 4 – Operação de convolução em exemplo [3].

É relevante notar que os filtros convolucionais são invariantes à translação da imagem, o que significa que produzem o mesmo resultado independentemente da posição do segmento na imagem. Além disso, operam com base nos princípios de localidade, analisando uma região da imagem por vez e combinando as análises para compreender a imagem como um todo [34].

2.3.2.3 Camada de Pooling

A camada de *pooling* é responsável por reduzir a dimensão dos dados de entrada, agrupando um conjunto de componentes vizinhos em um único componente. Em aplicações de processamento de imagens, essa camada seleciona um conjunto de pixels vizinhos por vez e, a partir deles, gera um único pixel representativo. Essa operação visa sintetizar e preservar ao máximo o significado geral da entrada, melhorando assim a capacidade do modelo de compreender as características relevantes dos dados [35].

Além de contribuir para a capacidade de generalização do modelo, essa redução de tamanho da entrada também auxilia na redução do custo computacional associado a uma CNN. Ao diminuir o tamanho da entrada, há uma consequente redução no número de parâmetros, refletindo positivamente nas camadas subsequentes do modelo [36].

Diversos tipos de *pooling* estão disponíveis, entre eles, o *max pooling* [37] e o *average pooling*, que são os mais comuns. No *max pooling*, o valor resultante é determinado como o maior elemento dentro de uma janela de itens, enquanto no *average pooling*, é calculada

a média de todos os elementos contidos na janela. Na Figura [5], é possível observar a aplicação de ambos os filtros de tamanho 2x2 em uma imagem 4x4.

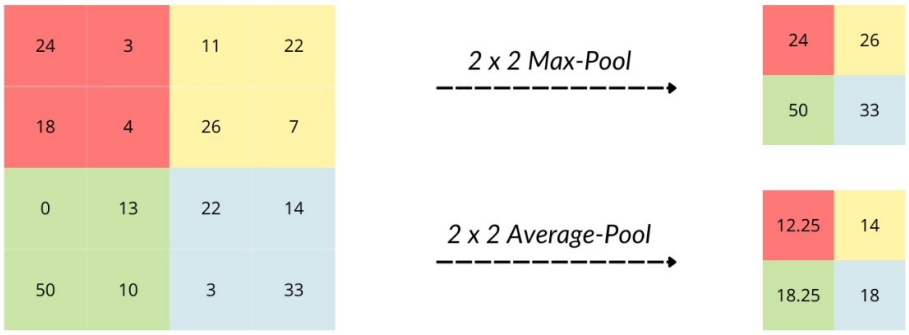

Figura 5 – Representação gráfica de um *max pooling* e *average pooling*. Elaboração: Autor

2.3.2.4 Arquitetura Convencional

As arquiteturas de CNN geralmente começam com uma camada de entrada, seguida por uma ou mais sequências de camadas convolucionais e de *pooling*. Depois dessas camadas, vêm as camadas totalmente conectadas. Tanto as camadas convolucionais quanto as camadas totalmente conectadas têm uma função de custo associada, e é comum encontrar operações de regularização nelas [38].

A Figura 6 é um exemplo de arquitetura de CNN que inclui uma camada de entrada, duas sequências de camadas convolucionais e de *pooling*, e, por último, uma camada totalmente conectada. Embora as arquiteturas possam variar em termos de ordem, tipos de camadas, funções de ativação e quantidade de camadas, a presença de camadas convolucionais e de *pooling* como as primeiras camadas é uma característica-chave de uma CNN.

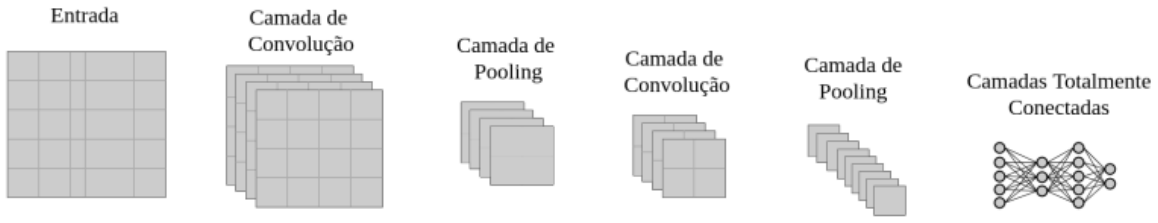

Figura 6 – Exemplo de arquitetura simples de CNN [3].

2.4 Reconhecimento de Objetos

O reconhecimento de objetos é uma disciplina da visão computacional que visa identificar e classificar objetos em imagens ou vídeos. Essa área desempenha um papel indispensável em uma variedade de aplicações, incluindo sistemas de segurança, veículos autônomos, assistência médica e educação [39]. O reconhecimento de objetos é desafiador devido à diversidade de objetos, variações de iluminação, oclusões e outras interferências presentes em imagens do mundo real.

As atividades de reconhecimento de objetos geralmente ocorrem em duas etapas distintas: a extração de características e a classificação, cada uma com suas próprias fases de operação, incluindo treinamento e reconhecimento. Durante a fase de treinamento do algoritmo, busca-se identificar padrões que representem os objetos de interesse. Esses padrões são discriminados através de um conjunto de características extraídas utilizando descritores, que podem ser locais ou globais. Esses descritores podem abranger uma variedade de informações, como aspectos geométricos, estatísticos, dados de cor, profundidade, borda, textura, entre outros [40].

Os descritores desempenham um papel essêncial na extração de características distintivas, conhecidas também como *features*, que permitem a identificação eficaz dos objetos de interesse. Eles capturam informações importantes sobre as características visuais dos objetos, como formas, texturas e padrões, e as representam de forma compacta e discriminativa. Essas características são então utilizadas pelos algoritmos de aprendizado de máquina para distinguir os objetos de interesse de outros elementos presentes na imagem [4] [41] .

No processo de reconhecimento de objetos, a complexidade pode aumentar significativamente quando o objeto de interesse deve ser distinguido de um fundo visualmente similar. Por exemplo, em imagens agrícolas, distinguir frutas do fundo composto por hastes e folhas é particularmente desafiador. Indira et al. [42] explicam que essa tarefa se torna ainda mais complicada devido às variações nas cores das frutas, à iluminação variável e à alta oclusão presentes nas imagens, o que dificulta a captação dos descritores.

Após a extração das características, elas são submetidas a um algoritmo de aprendizado de máquina que, utilizando algoritmos de agrupamento matemáticos, busca discernir as características que diferenciam o objeto de interesse dos demais na imagem. Isso resulta na geração de um modelo dos padrões encontrados.

A ilustração fornecida na Figura 7 apresenta um esquema geral de um algoritmo de reconhecimento de objetos.

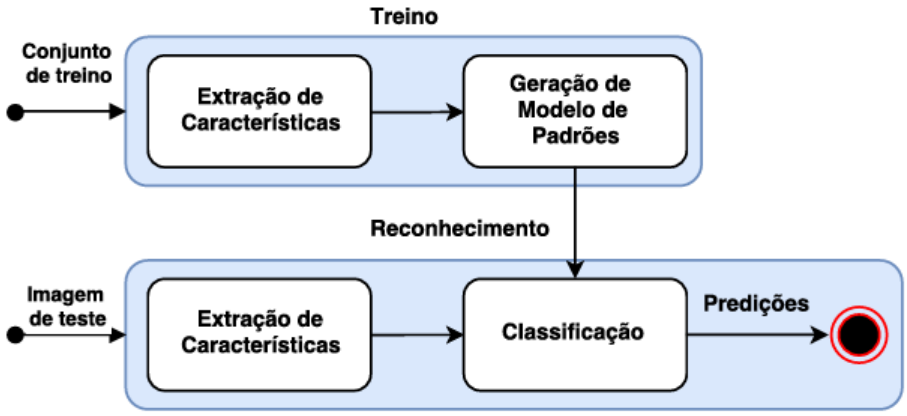

Figura 7 – Esquema geral para algoritmo de reconhecimento de objetos [4].

2.5 Trabalhos Relacionados

No estudo conduzido por Santos [43], uma abordagem utilizando redes neurais convolucionais foi empregada para a detecção de latas e garrafas em imagens. O treinamento do modelo envolveu mais de 300 imagens, incluindo tanto objetos de interesse quanto falsos positivos para garantir a robustez do modelo. Além disso, foram escolhidas 25 imagens de seis classes distintas, abrangendo desde barris até embalagens de plástico de refrigerantes, com o intuito de fornecer à rede um contexto diversificado para sua aprendizagem Os resultados obtidos demonstraram uma precisão média de 87,55% na detecção de latas e garrafas, contribuindo significativamente para o avanço da área de processamento de imagens e redes neurais artificiais.

No documento "Implementação de um Sistema de Reconhecimento de Objetos em Imagens" [4], foi explorada a implementação de técnicas de reconhecimento de objetos, utilizando o descritor Histograma de Orientações dos Gradientes (HOG) e Máquinas de Vetor de Suporte (SVM). O sistema desenvolvido teve como objetivo alcançar altas taxas de reconhecimento e robustez, que fosse capaz de lidar com variações como rotação, ruído, iluminação, entre outros fatores, sendo validado com sucesso em bases de dados como Caltech-101, MSRC v1 e Stanford Cars Dataset.

Em Kadhim et al. [44], foi implementado um sistema para reconhecimento e detecção de objetos visando auxiliar deficientes visuais em ambientes fechados. Para isso, foi utilizado o algoritmo "*You Only Look Once*" (YOLO), pré-treinado em conjunto com o dataset Common Objects in Context (COCO), para treinar uma rede neural convolucional (CNN). Esse sistema alcançou resultados promissores na detecção de 80 objetos comuns em diferentes ambientes. Os testes foram realizados em ambientes internos e externos, com diversas condições de iluminação, e mesmo assim o desempenho foi satisfatório, utilizando câmeras convencionais e uma Raspberry Pi. O YOLO foi escolhido para acelerar

o processo de detecção, proporcionando resultados rápidos e precisos.

Já no trabalho de Seema et al. [45], propuseram um sistema inteligente para auxiliar pessoas cegas na detecção de obstáculos. O sistema, embora focado na proteção da região próxima à cabeça, utilizou um *buzzer*, que é um dispositivo eletrônico que emite um som de alerta, e um vibrador como modos de alerta ao usuário. Essa abordagem se mostra útil para a detecção de obstáculos apenas ao nível da cabeça, sem, no entanto, reconhecer o tipo específico de obstáculo encontrado, funcionando mais como uma classificação de imagem e não um reconhecimento de objeto.

No trabalho de Cotrim [46], destaca-se a aplicação prática das redes neurais convolucionais no contexto dos processos de torra e forneamento de alimentos, como café e pães. A cor, como indicador crucial nessas transformações, é indispensável para o controle e classificação desses processos por meio de sistemas de visão computacional.

Para superar as limitações práticas dos modelos fenomenológicos tradicionais, o estudo introduziu técnicas de inteligência artificial (AI), focando especificamente nas CNN. A utilização de uma CNN com um número reduzido de camadas convolucionais resultou em economia significativa de memória, enquanto um sistema híbrido combinando CNN e máquinas de vetores de suporte (SVM) reduziu drasticamente o tempo de convergência.

Os resultados demonstraram que as CNN são capazes de extrair características de cores de forma seletiva, permitindo uma classificação precisa de amostras de café e pães. Com uma precisão superior a 98,0% na classificação de amostras de pães e 95% na classificação de amostras de café, as CNN superaram arquiteturas tradicionais. Além disso, as CNN mostraram-se capazes de estimar o tempo restante no processo de torra de café com um erro médio quadrático (RMSE) de apenas 0,4 minutos.

Esses resultados destacam o potencial das CNN como uma ferramenta prática e eficaz na modelagem do escurecimento não enzimático durante os processos de torra e forneamento de alimentos, com ampla aplicabilidade na indústria alimentícia.

O estudo de Ferreira [33] complementa esses avanços ao aplicar as Redes Neurais Convolucionais na detecção de ervas daninhas em lavouras de soja. Ao direcionar herbicidas específicos para diferentes tipos de ervas daninhas, classificando-as entre gramíneas e folhas largas, o estudo oferece soluções mais eficazes e sustentáveis para o manejo de culturas na agricultura de precisão.

Para comparar a eficácia das CNN, foram utilizados algoritmos como Máquina de Vetores de Suporte, AdaBoost e Random Forest, em conjunto com uma variedade de extratores de atributos, como forma, cor e textura. A precisão excepcionalmente alta alcançada na detecção das ervas daninhas, combinada com a capacidade de economizar tempo e recursos na aplicação de herbicidas, destaca a importância das CNN não apenas na indústria alimentícia, mas também na agricultura moderna, com uma média geral de

precisão superior a 99% em todas as imagens analisadas. Essas abordagens inovadoras representam um avanço significativo na aplicação de tecnologias de inteligência artificial para resolver desafios práticos em setores-chave da economia.

3 MATERIAIS E MÉTODOS

3.1 Visão Geral

O processo para o reconhecimento de objetos neste projeto segue uma série de etapas estruturadas. Inicia-se com a seleção de um conjunto de dados apropriado, com as anotações dos objetos, seguido pelo balanceamento das classes durante o pré-processamento dos dados.

Após isso, na etapa de pré-processamento das imagens, as mesmas são filtradas seguindo critérios que serão comentados nas seções posteriores, e também são normalizadas para melhorar sua qualidade e representação para entrada na rede neural.

Posteriormente, os dados são divididos em conjuntos de treino, teste e validação. O modelo de classificação é implementado e aplicado, e, por fim, o desempenho é avaliado utilizando métricas adequadas, onde os resultados são avaliados.

O fluxo geral do projeto pode ser visualizado na Figura 8.

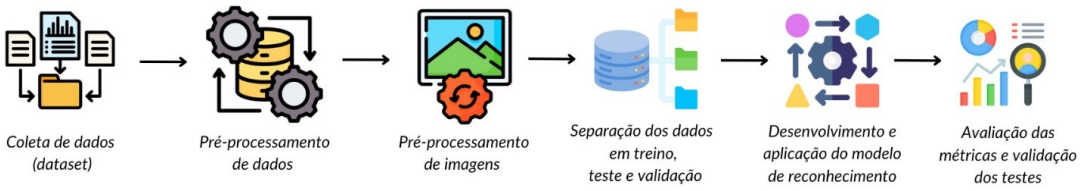

Figura 8 – Fluxo da visão geral.

3.2 Conjunto de Dados

O conjunto de dados utilizado neste trabalho foi o Microsoft Common Objects in COntext (COCO), uma base de dados de detecção, segmentação e descrição de objetos em larga escala, na versão de 2014 [47]. Esta base contém 90 categorias de objetos e um total de 330.000 imagens, contendo aproximadamente 2.500.000 instâncias de objetos.

Cada objeto reconhecido dentro das imagens do conjunto de dados é acompanhado por um arquivo JSON que contém uma variedade de informações essenciais. Esses dados incluem a área ocupada pelo objeto na imagem, que é uma medida importante para entender o tamanho e a relevância do objeto em relação ao restante da cena. Além disso, são fornecidas as coordenadas do retângulo delimitador (*bounding box*) que circunscreve

o objeto, o que permite uma representação mais simplificada e eficiente da sua posição e tamanho na imagem.

Outra informação importante é a categoria, ou rótulo, à qual o objeto pertence. Cada objeto é atribuído a uma categoria específica, definida previamente no conjunto de dados, o que facilita a análise e interpretação dos resultados. Além disso, são fornecidas as coordenadas do polígono que descreve a forma precisa do objeto. Embora essas informações detalhadas do polígono não tenham sido utilizadas no contexto deste trabalho, elas são valiosas para aplicações que exigem uma segmentação mais precisa e detalhada dos objetos na imagem.

A escolha do COCO foi motivada por diversas razões. Primeiramente, o conjunto de dados oferece uma ampla variedade de categorias de objetos, cobrindo uma vasta gama de contextos e cenários. Essa diversidade é substancial para garantir que os modelos de reconhecimento de objetos sejam robustos e generalizados o suficiente para lidar com diferentes situações do mundo real. Além disso, o COCO fornece anotações detalhadas para cada imagem, incluindo informações como a área ocupada pelo objeto, essenciais para o desenvolvimento de algoritmos de detecção de objetos. A disponibilidade e o uso generalizado do COCO na comunidade científica também foram fatores importantes na escolha, proporcionando recursos e orientações para o uso e análise dos dados. Esses aspectos combinados tornam o conjunto de dados da Microsoft uma escolha sólida e abrangente para este estudo de reconhecimento de objetos.

No contexto deste trabalho, o foco principal está nas coordenadas do *bounding box* e na categoria de cada objeto, que serão exploradas em detalhes para treinar e avaliar modelos de reconhecimento de objetos. Esses dados são precisos para o desenvolvimento e avaliação de algoritmos de reconhecimento de objetos, fornecendo informações de grande significância para o treinamento e aprimoramento dos modelos. Um exemplo de *bounding box* está representado na Figura 9, onde a classe reconhecida é "pessoa".

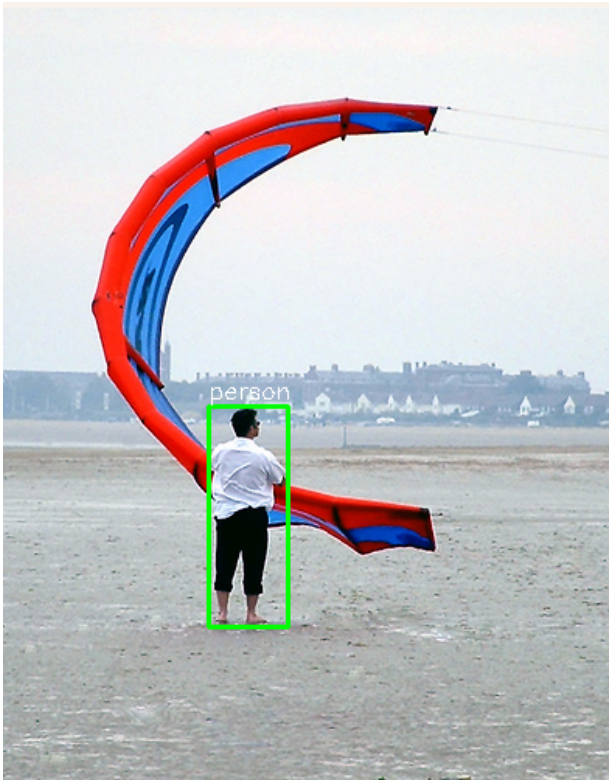

Figura 9 – Imagem rotulada com sua classe e com *bounding box* desenhado de acordo com a anotação do conjunto de dados. Elaboração: Autor

Enquanto na Figura 10 pode-se visualizar exemplos de imagens de 5 classes diferentes das 90 disponíveis no dataset utilizado.

Figura 10 – Exemplos de imagem por classe do COCO. Elaboração: Autor

3.3 Métricas de Avaliação

Uma parte importante da análise de experimentos de aprendizado de máquina é a avaliação quantitativa do desempenho do modelo. Nesta seção, discutiremos várias métricas que ajudam a medir o comportamento e a eficácia de modelos de redes neurais. Cada métrica desempenha um papel específico; permite uma compreensão mais profunda de como o modelo se ajusta aos dados de treinamento e teste. Estas métricas ajudam a direcionar o refinamento e a otimização do modelo, desde a avaliação da precisão global até

a análise mais detalhada da capacidade do modelo de identificar corretamente diferentes classes [48].

A seguir, abordaremos em detalhes cada uma dessas métricas, enfatizando seus propósitos específicos e o valor que agregam à avaliação de redes neurais em diferentes situações.

3.3.1 Matriz de Confusão

A matriz de confusão é um instrumento essencial para avaliar modelos de classificação. Ela permite visualizar o desempenho de um algoritmo ao comparar as classes reais e previstas. Nessa representação, as contagens de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos são organizadas, destacando a diagonal principal, onde as previsões corretas do modelo são apresentadas - isto é, as instâncias em que as classes reais e previstas coincidem. Por outro lado, células fora da diagonal principal denotam erros de classificação [49]. Por exemplo, falsos positivos ocorrem quando o modelo prevê 'é gato', mas a classe real é 'não é gato', enquanto falsos negativos ocorrem quando o modelo prevê 'não é gato', mas a classe real é 'é gato'. Ilustrando isso, na Figura 11, uma matriz de confusão com duas classes ('é gato' e 'não é gato') é exibida, destacando esses exemplos. Essa visualização oferece uma compreensão abrangente de como o modelo classifica as amostras, proporcionando informações valiosas sobre seu desempenho em várias classes.

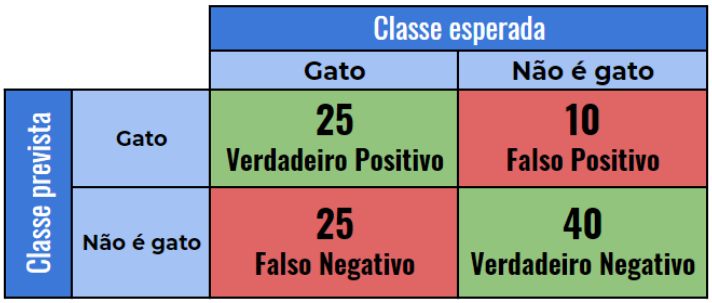

Figura 11 – Matriz de confusão com classes.

3.3.2 Acurácia

A acurácia é uma métrica comumente usada para avaliar o desempenho de um modelo de classificação em sua totalidade. Ela mostra a proporção de previsões corretas em relação ao número total de previsões que o modelo fez. Em outras palavras, a acurácia representa a capacidade do modelo de classificar amostras corretamente em todas as classes [50]. No entanto, em casos de desbalanceamento de classes, a acurácia pode ser enganosa, pois um modelo pode ter alto desempenho em classes majoritárias e baixo

desempenho em classes minoritárias, distorcendo a real eficácia do modelo [51]. O cálculo da acurácia é realizado por meio da Equação 3.1 [52].

$$\text{Acc} = \frac{VP + VN}{VP + FP + FN + VN} \tag{3.1}$$

3.3.3 Precisão

A precisão é uma medida que mede a proporção de amostras positivas verdadeiras (aquelas que foram corretamente classificadas como positivas) em relação ao número total de amostras que foram classificadas como positivas pelo modelo. Em outras palavras, a precisão indica a qualidade das previsões positivas do modelo. Uma alta precisão indica que o modelo tem uma baixa taxa de falsos positivos, ou seja, poucas amostras negativas são erroneamente classificadas como positivas. Essa medida é calculada pela Equação 3.2 [52].

$$\text{Prec} = \frac{VP}{VP + FP} \tag{3.2}$$

3.3.4 Revocação (Recall)

A revocação, também chamada de recall, é uma medida que determina a proporção de amostras positivas verdadeiras em relação ao total de amostras que realmente pertencem à classe positiva. Em outras palavras, a revocação mostra a capacidade do modelo de identificar todas as amostras positivas corretamente. A alta revocação indica que o modelo tem uma baixa taxa de falsos negativos, ou seja, poucas amostras positivas são erroneamente classificadas como negativas, e para sabermos calculamos utilizando a Equação 3.3 [52].

$$\text{Rev} = \frac{VP}{VP + FN} \tag{3.3}$$

3.3.5 F1-Score

O *F1-score* é uma métrica que combina a precisão e a revocação em uma única medida, calculada como a média harmônica dessas duas métricas. Quando as classes não estão equilibradas ou quando a precisão e a revocação são igualmente importantes, o *F1-score* é especialmente útil. Em geral, um alto *F1-Score* indica um bom equilíbrio de precisão e revocação. Tem-se a combinação da métrica na Equação 3.4 [52].

$$\text{F1 Score} = 2 * \frac{Prec * Rev}{Prec + Rev} \tag{3.4}$$

3.4 Experimento

3.4.1 Seleção de imagens

Na seção 3.2, descreve-se o conjunto de dados utilizado neste estudo, o Microsoft Common Objects in COntext (COCO). Durante a análise do conjunto de dados, foi identificado um desbalanceamento nas classes, com algumas categorias apresentando um número significativamente maior de instâncias do que outras.

O desbalanceamento nas classes pode levar a uma performance inferior do modelo de aprendizado de máquina, especialmente em tarefas de classificação. Estudos como o de He et al. [51] demonstraram que modelos treinados em conjuntos de dados desbalanceados tendem a favorecer as classes majoritárias, resultando em uma baixa capacidade de generalização para as classes minoritárias. Essa falta de balanceamento pode levar a uma baixa acurácia e a uma alta taxa de falsos positivos ou falsos negativos, prejudicando a eficácia do modelo em aplicações do mundo real.

Diante desse desbalanceamento, com o objetivo de garantir uma distribuição mais equilibrada das classes durante o treinamento da rede neural, foi adotado um processo de pré-processamento do conjunto de dados COCO, consistindo em uma seleção criteriosa das imagens.

Inicialmente, o dataset original, tanto de treinamento quanto de validação, foi lido e filtrado, selecionando-se 4000 imagens por classe. Essas imagens foram então separadas em pastas de acordo com suas respectivas classes, estabelecendo-se um limite superior para garantir uma distribuição mais equilibrada das classes durante o treinamento da rede neural. As classes escolhidas e filtradas para o treinamento neste trabalho em específico foram "mesa de jantar", "cadeira" e "pessoa", escolhidas devido à sua relevância para a aplicação proposta. Essas classes representam objetos comuns encontrados em ambientes domésticos e sociais. Além disso, observou-se uma distribuição mais equilibrada entre essas classes no conjunto de dados.

Durante esse processo de seleção, os *bounding boxes* de cada imagem, que delimitam o objeto de interesse, foram analisados para verificar a área do objeto em relação à área total da imagem. Caso essa proporção fosse menor que 20%, a imagem correspondente seria descartada. Essa medida foi adotada para remover objetos muito pequenos que poderiam ter uma baixa resolução após o recorte citado, prejudicando assim a extração das *features*.

A decisão de estabelecer o limite em 20% foi baseada em uma análise cuidadosa das imagens e dos objetivos do projeto. Foram realizados alguns testes preliminares e observamos que, ao definir o limite em 30%, muitas imagens foram descartadas, resultando em uma redução significativa no tamanho do conjunto de dados e potencialmente comprometendo a diversidade e representatividade das amostras. Por outro lado, ao reduzir

o limite para menos de 20%, ainda observamos a presença de objetos muito pequenos que poderiam ter uma baixa resolução após o recorte, afetando negativamente a extração das características. Portanto, optamos por estabelecer o limite em 20%, pois proporciona um equilíbrio entre a remoção de objetos muito pequenos e a preservação do tamanho e qualidade do conjunto de dados.

Na Figura 12, observa-se que o objeto reconhecido, classificado como "Sports Ball", possui uma área significativamente menor.

Figura 12 – Exemplo de imagem com objeto pequeno removido dos dados utilizados para o treinamento da rede neural artificial. Elaboração: Autor

Em seguida, foi estabelecido um limite superior adicional para a quantidade de imagens por classe, tanto no conjunto de dados de treinamento quanto no conjunto de validação, fixando o valor de 1000 imagens por classe. Essa medida foi tomada com o objetivo de agilizar as comparações de resultados deste trabalho, possibilitando o uso de conjuntos de dados com diferentes quantidades de imagens de maneira mais eficiente no treinamento do modelo. Isso evitou a necessidade de acessar o conjunto de dados completo e realizar o balanceamento novamente, o que contribuiu para uma execução mais rápida e simplificada dos experimentos.

Posteriormente, após a seleção e carregamento de 800 imagens por classe, juntamente com suas respectivas anotações, foi aplicado o *data augmentation* (DA) para expandir o conjunto de dados. A importância do uso de *data augmentation* no treinamento do modelo de reconhecimento foi amplamente documentada na literatura. Estudos como o de Shorten et al. [53] demonstraram que a aplicação de técnicas de DA, como rotação, translação, espelhamento e zoom, pode melhorar significativamente a generalização do modelo, reduzindo o risco de *overfitting* e aumentando a robustez do modelo em relação a variações nos dados de entrada. Além disso, o aumento da diversidade dos dados pode ajudar a evitar o viés do modelo em direção às classes majoritárias, resultando em um desempenho mais equilibrado em todas as classes.

Utilizou-se a biblioteca *ImageDataGenerator* do TensorFlow, aproveitando suas funcionalidades, como rotação, deslocamento horizontal e vertical, *zoom*, espelhamento horizontal e preenchimento de *pixels* novos. Essas operações foram implementadas em Python. Como resultado desse processo, cada classe foi ampliada para conter 4000 imagens.

Em sequência, o conjunto de imagens atribuídas ao conjunto de treinamento foi dividido em conjuntos de treino e teste. Esse processo foi realizado utilizando a biblioteca scikit-learn (sklearn), que oferece funcionalidades para divisão de dados em conjuntos de treinamento e teste de forma aleatória e estratificada. A divisão foi realizada com uma proporção de 70% para o conjunto de treinamento e 30% para o conjunto de teste, oferecendo uma quantidade suficiente de dados para o treinamento do modelo, permitindo que ele aprenda padrões nos dados de treinamento, enquanto ainda reserva uma parte significativa dos dados para avaliar a capacidade de generalização do modelo em dados não vistos.

Dessa forma, o processo de divisão resultou em um conjunto de dados balanceado. Assim, cada classe conseguiria contribuir sem muita discrepância para o treinamento da rede neural, garantindo uma representação equitativa de todas as classes durante o processo de aprendizado.

3.4.2 Pré-processamento

O pré-processamento de imagens é necessário para preparar os dados de entrada para o treinamento do modelo de reconhecimento de objetos e pode incluir diversos processos, como estruturação de dados, filtragem de base e clusterização/segmentação [54]. As imagens foram otimizadas e preparadas para uso pelo modelo por meio das seguintes técnicas:

O primeiro passo foi o recorte dos objetos, onde as regiões de interesse foram preservadas por meio do recorte das imagens com base na *bounding box* fornecida nas anotações do dataset COCO. Isso permite que o modelo se concentre nas partes relevantes das imagens durante o treinamento, melhorando sua capacidade de reconhecimento.

Após o recorte dos objetos, a função de nitidez (*sharpen*) também foi aplicada às imagens utilizando a biblioteca Pillow em Python para isso. As imagens originalmente recortadas foram filtradas com o uso de nitidez para realçar os detalhes e melhorar a definição das características relevantes. Esse processo representado na Figura 13 visa aprimorar ainda mais a qualidade das imagens antes de alimentá-las ao modelo de reconhecimento. Após a aplicação do filtro, a imagem foi convertida para um *array numpy* para ser processada posteriormente pelo modelo de aprendizado de máquina. Essa conversão é essencial para a manipulação eficiente de dados de imagem em Python, permitindo operações matemáticas e de matriz de maneira otimizada [55] [56].

Figura 13 – Antes e depois da aplicação do filtro *sharpen*.

Em seguida, as imagens foram redimensionadas para um tamanho específico de 128x128 pixels, utilizando a função resize da biblioteca PIL (Python Imaging Library). Esse redimensionamento padroniza o tamanho das imagens de entrada, garantindo uma entrada consistente para o modelo. Isso simplifica o processo de treinamento e torna a inferência mais eficiente.

Já os valores dos pixels das imagens foram normalizados para o intervalo [0, 1], dividindo cada valor por 255. Essa normalização é uma prática comum em tarefas de processamento de imagem e ajuda no treinamento eficiente do modelo, garantindo que os gradientes durante o treinamento permaneçam em uma escala adequada [57].

Também realizamos o embaralhamento dos dados, que foi um passo significativo adicional. As imagens e os rótulos correspondentes foram combinados em pares, embaralhados aleatoriamente e depois separados de volta em duas listas distintas. Essa fase foi importante para evitar que o modelo aprenda sequências ou padrões específicos indesejados durante o treinamento. O embaralhamento dos dados garante que os exemplos de treinamento sejam distribuídos aleatoriamente, o que torna o processo de aprendizado mais sólido e eficaz. O modelo é incentivado a aprender as características pertinentes dos objetos em vez de depender de ordens específicas dos dados de entrada. A literatura indica que o embaralhamento dos dados pode levar a um melhor desempenho dos modelos, prevenindo o *overfitting* e promovendo a generalização [10].

3.4.3 Implementação da Rede Neural e Treinamento do Modelo

Foram utilizadas as bibliotecas Keras [58] e TensorFlow [59] para a implementação da rede neural. Essas ferramentas proporcionam uma estrutura flexível e eficiente para construir e treinar modelos de aprendizado profundo. Inicialmente, os dados foram preparados carregando as imagens e suas anotações correspondentes de arquivos JSON.

A arquitetura da rede neural foi definida em camadas sequenciais, utilizando a abordagem de modelagem do tipo *sequential* do Keras. Primeiramente, as camadas convolucionais foram empregadas. Essas camadas são responsáveis por extrair características importantes das imagens, aplicando operações de convolução que detectam padrões locais em regiões específicas da imagem. A estratégia para criação da arquitetura consistiu em começar com camadas convolucionais com um número menor de filtros para captar informações mais básicas das imagens. Conforme a rede se aprofundava, camadas convolucionais com um número maior de filtros eram adicionadas para capturar informações mais complexas e abstratas. Cada camada convolucional é seguida por uma camada de normalização em lote para estabilizar o treinamento e uma camada de *max pooling* para reduzir a dimensionalidade dos mapas de características.

Para reduzir o *overfitting* durante o treinamento, foram inseridas camadas de *dropout* entre as camadas convolucionais. Essas camadas descartam aleatoriamente uma fração das ativações da camada anterior, o que ajuda a evitar que a rede memorize o conjunto de treinamento e melhora sua capacidade de generalização para dados não vistos [60].

Após as camadas convolucionais, os mapas de características 2D foram transformados em um vetor unidimensional usando uma camada de *flatten*. Em seguida, foram adicionadas camadas densas (totalmente conectadas) para realizar a classificação final. A função de ativação *relu* foi usada nas camadas convolucionais, enquanto a função de ativação *softmax* foi utilizada na camada de saída para gerar probabilidades para cada classe. Assim, o modelo detalhado da rede neural e suas camadas está representado na Tabela [1].

O modelo foi compilado especificando o otimizador *adam* e a função de perda *sparse_categorical_crossentropy*, adequada para problemas de classificação com várias classes. Durante o treinamento, o modelo foi ajustado utilizando um conjunto de dados de treinamento por 15 épocas. Uma época, em termos de treinamento de redes neurais, refere-se a uma única passagem completa de todo o conjunto de dados de treinamento pelo modelo. Durante cada época, o modelo ajusta seus pesos e parâmetros internos com base nos dados de treinamento, com o objetivo de minimizar a função de perda e melhorar o desempenho geral do modelo. A escolha de 15 épocas foi baseada em testes preliminares, onde se observou que o modelo alcançou sua melhor acurácia com este número de épocas, equilibrando entre o sobreajuste e o subajuste. Estudos indicam que monitorar a acurácia e a perda durante o treinamento é uma prática comum para determinar o número ideal de épocas [10].

Além disso, foi utilizado um tamanho de lote de 32, o que significa que o modelo atualiza seus parâmetros após processar 32 amostras de treinamento em cada iteração. Este valor é uma escolha comum que equilibra a eficiência computacional e a estabilidade do treinamento [61]. Um conjunto de validação foi utilizado para monitorar o desempenho

Layer (type)	**Output Shape**	**Param #**
conv2d (Conv2D)	(None, 126, 126, 8)	224
batch_normalization (BatchNormalization)	(None, 126, 126, 8)	32
max_pooling2d (MaxPooling2D)	(None, 63, 63, 8)	0
dropout (Dropout)	(None, 63, 63, 8)	0
conv2d_1 (Conv2D)	(None, 61, 61, 16)	1,168
batch_normalization_1 (BatchNormalization)	(None, 61, 61, 16)	64
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 16)	0
dropout_1 (Dropout)	(None, 30, 30, 16)	0
conv2d_2 (Conv2D)	(None, 28, 28, 32)	4,640
batch_normalization_2 (BatchNormalization)	(None, 28, 28, 32)	128
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 32)	0
dropout_2 (Dropout)	(None, 14, 14, 32)	0
conv2d_3 (Conv2D)	(None, 12, 12, 64)	18,496
batch_normalization_3 (BatchNormalization)	(None, 12, 12, 64)	256
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 512)	1,180,160
dense_1 (Dense)	(None, 3)	1,539

Tabela 1 – Detalhes da Arquitetura da Rede Neural

do modelo em dados não vistos durante o treinamento e evitar o sobreajuste.

O treinamento do modelo foi realizado em um computador com as seguintes especificações de hardware: um laptop Dell G3 equipado com um processador Intel Core i7 de 11ª geração, 16GB de RAM e uma placa de vídeo dedicada NVIDIA GeForce GTX 1060 Ti.

Após o treinamento, o modelo foi avaliado utilizando um conjunto de teste separado para determinar sua acurácia final. Além disso, foram calculadas as métricas de precisão, *recall* e *F1 Score* para avaliar o desempenho do modelo de forma mais abrangente. Para visualizar o progresso do modelo durante o treinamento, foram plotadas curvas de perda e acurácia em relação ao número de épocas. Além disso, a matriz de confusão de teste foi gerada para fornecer uma representação visual do desempenho do modelo na classificação de diferentes classes no conjunto de teste. Essa matriz permite analisar os acertos e erros do modelo em cada classe, ajudando a identificar possíveis padrões de confusão.

4 RESULTADOS

4.1 Apresentação dos resultados

O modelo foi treinado ao longo de 15 épocas, com avaliações realizadas durante o treinamento. A Figura 14 mostra o gráfico de acurácia de treino e validação ao longo das épocas, enquanto a Figura 15 representa o gráfico de perda de treino e validação.

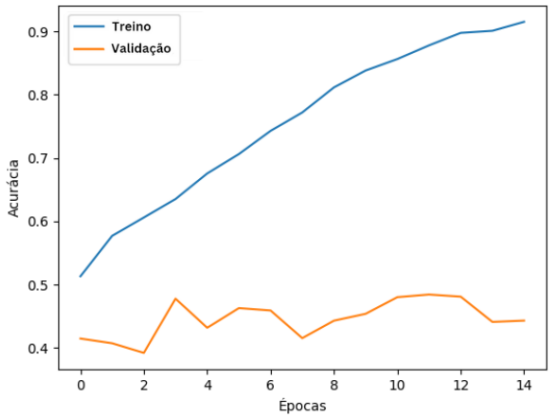

Figura 14 – Gráfico de acurácia de treino e validação pelas épocas.

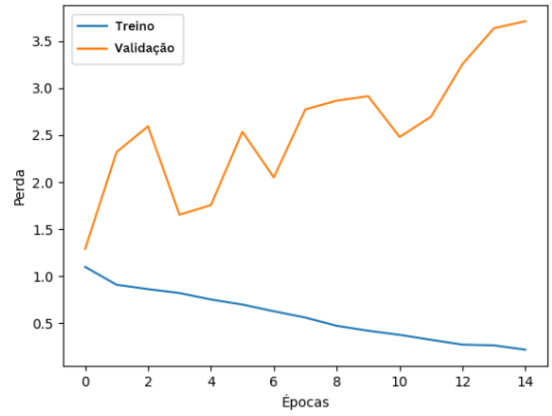

Figura 15 – Gráfico de perda de treino e validação pelas épocas.

Observando a Figura 14, nota-se que a acurácia do modelo aumentou gradualmente durante as épocas no conjunto de treinamento, mas não no conjunto de validação. Inicialmente, a acurácia no conjunto para treino era alta, com valores em torno de 91,61%, indicando que o modelo estava aprendendo bem os padrões presentes nos dados de treinamento. No entanto, a acurácia no conjunto de validação era consideravelmente mais

baixa, em torno de 44,33%, sugerindo que o modelo estava encontrando dificuldades para generalizar para dados não vistos.

A discrepância entre as perdas observadas nos conjuntos de treinamento e validação permite inferir questões relativas o desempenho do modelo. Enquanto a perda no conjunto de treinamento é relativamente baixa, aproximadamente 0,2116, indicando uma boa adaptação aos dados de treinamento, a perda no conjunto de validação é significativamente maior, em torno de 3,7116. Essa disparidade sugere que o modelo pode estar capturando ruídos e padrões específicos das imagens durante o treinamento, o que resulta em um desempenho inferior quando aplicado a dados de validação.

Esses resultados reforçam a possibilidade de *overfitting*, onde o modelo se ajusta demasiadamente aos dados de treinamento, perdendo a capacidade de generalizar para novos exemplos.

Posteriormente, foi utilizado o conjunto de teste para avaliar o desempenho do modelo, o que resultou na matriz de confusão mostrada na Figura 16.

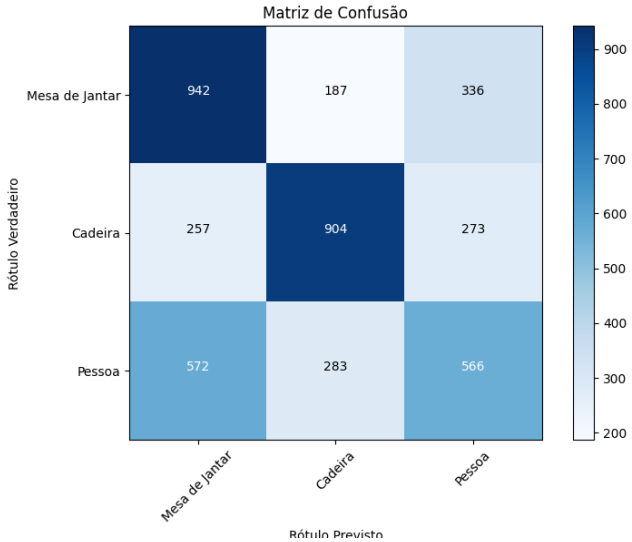

Figura 16 – Matriz de confusão de teste do modelo.

Ao analisar a distribuição de previsões erradas na matriz de confusão, observamos um padrão na confusão das imagens da classe "Mesa de Jantar". É possível observar que, em muitos casos, uma mesa de jantar está frequentemente próxima a uma cadeira, que por sua vez pode estar ocupada por uma pessoa. No entanto, ao recortar o objeto utilizando *bounding boxes* na imagem, sua delimitação não é feita por meio de pontos de polígonos detalhados. Essa imprecisão na delimitação pode levar o modelo a interpretar erroneamente as mesas como outros objetos presentes no mesmo recorte, resultando em confusões na classificação.

A partir da matriz de confusão, as métricas de avaliação de precisão, revocação,

F1-Score e acurácia, foram calculadas e estão apresentadas na Tabela 2.

Métrica	Valor
Precisão	0,557
Revocação	0,558
F1-Score	0,555
Acurácia	0,558

Tabela 2 – Métricas de Teste

Os dados revelam uma performance que, embora não seja ideal, oferece informações valiosas sobre as capacidades e limitações do modelo em relação às classes de interesse. Com uma precisão em torno de 0,557, aproximadamente 55,7% das instâncias classificadas como positivas são de fato positivas, o que sugere uma habilidade do modelo em realizar previsões, embora com uma alta margem de erro.

No entanto, a revocação de aproximadamente 0,558 indica uma dificuldade significativa em capturar todas as instâncias positivas existentes no conjunto de teste. Isso sugere que o modelo pode estar tendo dificuldades em identificar corretamente exemplos importantes de cada classe, o que pode impactar sua eficácia em situações práticas.

Apesar de um *F1-Score* moderado de aproximadamente 0,555, refletindo um equilíbrio delicado entre precisão e revocação, ainda há espaço para melhorias para alcançar um desempenho mais consistente.

A acurácia em torno de 0,558, embora indique uma capacidade razoável de classificação, não deve ser considerada como um indicativo de desempenho totalmente satisfatório. O modelo classifica corretamente cerca de 55,8% das instâncias no conjunto de teste, o que deve ser aprimorado para uma aplicação mais confiável.

Ao analisar essas métricas em conjunto, identificamos algumas áreas que podem ser aprimoradas no modelo, como a precisão na identificação das instâncias positivas, a revocação para capturar todas as instâncias relevantes e o equilíbrio entre precisão e revocação representado pelo *F1-Score*. Portanto, é essencial continuar revisando e refinando o modelo em relação a esses aspectos, além de explorar possíveis abordagens alternativas para otimizar seu desempenho em cenários práticos.

Adicionalmente, realizamos testes em diferentes números de épocas (5, 15, 25 e 40) para observar a variação das métricas. Os resultados desses testes são apresentados na Tabela 3 e visualizados na Figura 17.

Épocas	Acurácia	Precisão	Revocação	F1-Score
5	0,406	0,475	0,406	0,319
15	**0,558**	**0,557**	**0,558**	**0,555**
25	0,425	0,542	0,425	0,349
40	0,513	0,540	0,513	0,460

Tabela 3 – Comparação das métricas por número de épocas.

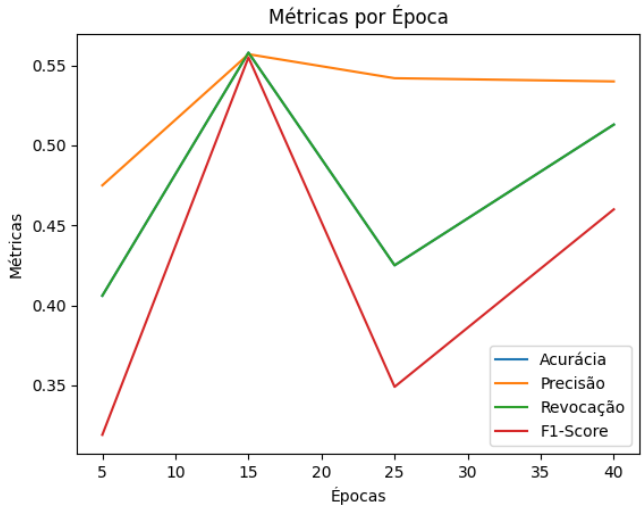

Figura 17 – Gráfico de variação das métricas por número de épocas.

Pode-se observar que variação significativa no desempenho do modelo em diferentes estágios do treinamento. Destacamos que o modelo treinado com 15 épocas obteve os melhores resultados em todas as métricas avaliadas.

Inicialmente, o modelo treinado com apenas 5 épocas apresentou uma acurácia de 0,406, precisão de 0,475, revocação de 0,406 e *F1-Score* de 0,319. Esses valores indicam um desempenho relativamente baixo, sugerindo que o modelo ainda não teve tempo suficiente para aprender os padrões presentes nos dados de treinamento.

À medida que o número de épocas aumentou para 15, observamos uma melhoria significativa em todas as métricas. A acurácia atingiu o valor mais alto de 0,558, indicando uma capacidade considerável do modelo de fazer previsões corretas no conjunto de teste em comparação à números menores de épocas. Além disso, tanto a precisão, revocação quanto o *F1-Score* aumentaram para valores em torno de 0,557 e 0,558, respectivamente, sugerindo uma melhoria geral no desempenho do modelo.

Por outro lado, ao aumentar o número de épocas para 25 e 40, observamos uma queda no desempenho do modelo em todas as métricas. Embora a acurácia tenha se mantido relativamente estável em torno de 0,425 e 0,513, a precisão, revocação e *F1-Score* diminuíram, indicando uma possível ocorrência de *overfitting*, onde o modelo se ajusta

demais aos dados de treinamento e não generaliza bem para novos dados, o que mostra que nem sempre quanto maior a quantidade de épocas melhor para o aprendizado do modelo.

Após 15 épocas, o algoritmo pode ter enfrentado o fenômeno conhecido como "degradação do gradiente". Esse fenômeno ocorre quando o algoritmo de otimização, durante o treinamento, não consegue encontrar uma direção adequada para atualizar os pesos do modelo, resultando em uma desaceleração na convergência ou até mesmo na divergência do processo. Este cenário é bastante comum em redes neurais profundas, onde os gradientes podem se tornar muito pequenos ou muito grandes à medida que são propagados de volta através das camadas da rede.

Para mitigar esse problema, uma alternativa é empregar técnicas como o ajuste adaptativo da taxa de aprendizado, utilizar algoritmos de otimização mais avançados, como Adam ou RMSprop, ou normalizar os gradientes durante o treinamento, a fim de evitar explosões ou desvanecimento dos gradientes.

A Figura 18 ilustra dois exemplos de reconhecimento de objetos pelo modelo treinado. Na primeira imagem, o objeto foi corretamente reconhecido, enquanto na segunda imagem, o objeto foi erroneamente identificado como uma pessoa pelo algoritmo desenvolvido.

Figura 18 – Exemplos de reconhecimento de objetos pelo modelo treinado.

4.2 Análise de Estratégias de Melhoria Durante o Desenvolvimento do Modelo

Nesta seção, uma análise comparativa dos resultados obtidos por meio de diferentes abordagens de processamento de dados e imagens é apresentada.

4.2.1 Conjunto de dados desbalanceado

Ao analisar os resultados do modelo treinado com dados desbalanceados, devemos ressaltar a relevância da análise da acurácia por classe. Mesmo após o treinamento com dados desbalanceados, fica evidente que a acurácia varia significativamente entre as diferentes categorias, observando a Tabela 4. Além disso, a acurácia global do modelo no conjunto de teste foi de aproximadamente 26,7%, evidenciando os desafios enfrentados devido ao desbalanceamento dos dados.

Classe	Amostras Original	Amostras Desbalanceado	Acurácia por Classe
Mesa de Jantar	800	684	47,3%
Pessoa	800	551	13,9%
Cadeira	800	160	66,7%

Tabela 4 – Comparação entre conjuntos de dados original e desbalanceado com acurácia por classe no desbalanceamento.

Por exemplo, a classe "Mesa de Jantar" apresenta uma acurácia de 47,3%, enquanto a classe "Pessoa" possui uma acurácia muito inferior, de apenas 13,9%. Essa diferença ressalta a dificuldade do modelo em reconhecer corretamente certas classes quando há um desbalanceamento no conjunto de dados.

Além disso, a classe "Cadeira" mostra uma acurácia relativamente alta de 66,7%. Porém é importante considerar que essa alta acurácia pode ser enganosa devido ao menor número de amostras disponíveis para essa classe no conjunto desbalanceado.

Essas discrepâncias na acurácia por classe destacam a necessidade de uma análise detalhada do desempenho do modelo em cada categoria, especialmente em situações de desbalanceamento de dados. Importante ressaltar também a métrica *F1-Score*, que, nesse caso, ficou em 22,6%.

4.2.2 Imagens sem recorte por bbox

Sem utilizar o recorte do *bounding box* na imagem extraindo apenas o objeto para ser alimentado no modelo, teve-se uma queda no desempenho do modelo, que se perdeu com as informações adicionais poluindo as características do objeto. A acurácia que estava em 56% foi para 36,3% após a remoção desse pré-processamento identificado pela Figura 19.

Para avaliar o desempenho do modelo, foram comparadas as métricas de validação e teste, conforme apresentado na Tabela 1. Nota-se que a acurácia no conjunto de teste é de 36,3%, uma queda significativa em relação à acurácia durante a validação. Essa diferença de 7,68% ressalta a importância de avaliar o modelo em dados não vistos durante o treinamento para garantir sua capacidade de generalização. Além disso, métricas

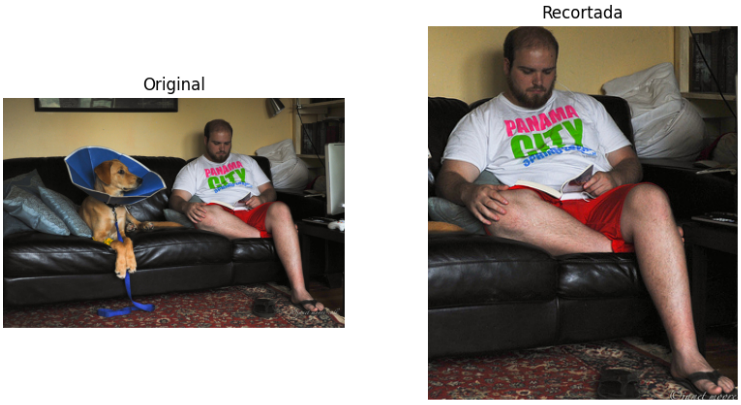

Figura 19 – Comparação de imagem original e recortada.

como precisão, revocação e *F1-Score* também mostram uma redução no conjunto de teste, sugerindo a possibilidade de *overfitting* do modelo aos dados de treinamento.

Métrica	**Validação**	**Teste**	**Diferença (%)**
Acurácia	43,98	36,3	-7,68
Precisão	48,8	44,9	-3,9
Revocação	44,3	36,4	-7,9
F1-Score	45,3	28,3	-17,0

Tabela 5 – Comparação das métricas de validação e teste sobre o *bbox*.

4.2.3 Conjunto de dados sem ampliação

Nesta seção, temos as diferenças de desempenho entre o modelo desenvolvido quando aplicado ao conjunto de dados original e ao conjunto com aumento de dados (*data augmentation*). Na Figura 21, apresentamos exemplos de filtros aplicados para o aumento de dados, incluindo rotação, deslocamento horizontal e vertical, zoom, inversão horizontal e preenchimento para geração de novas imagens.

Figura 20 – Exemplos de filtros aplicados para aumento de dados.

Ao removermos o *data augmentation*, o conjunto de treinamento se reduziu para 1680 imagens no total, distribuídas entre as três classes. Isso resultou em uma queda na

acurácia, de 0,558 para 0,451. As métricas de precisão, revocação e **F1-Score** também foram afetadas, apresentando os valores da Tabela 6.

Métrica	Valor
Precisão	0,506
Revocação	0,451
F1-Score	0,427

Tabela 6 – Métricas do modelo sem *data augmentation*.

A redução no desempenho pode ser atribuída à diminuição do conjunto de dados, que não capturou as variações presentes nas imagens do dataset aumentado. Quando testado em imagens inéditas, onde não estão presentes os mesmos ângulos, zoom ou posicionamento horizontal ou vertical das imagens usadas no treinamento, o modelo mostrou uma diminuição na eficácia, como observado.

5 CONCLUSÃO

Este trabalho buscou desenvolver e avaliar um modelo de reconhecimento de objetos utilizando técnicas de aprendizado profundo. Para isso, foram utilizados dados do conjunto COCO, pré-processados e alimentados em uma arquitetura de rede neural convolucional. O modelo foi treinado ao longo de 15 épocas e avaliado com base em métricas de desempenho, incluindo acurácia, precisão, revocação e *F1-Score*.

Os resultados obtidos não alcançaram o nível de desempenho esperado. A acurácia em torno de 55.8% e outras métricas moderadas indicam limitações significativas no modelo. Ademais, observou-se uma disparidade entre as métricas nos conjuntos de treinamento e validação, sugerindo possíveis problemas de *overfitting*, comprometendo a generalização do modelo. A análise da matriz de confusão revelou certos padrões de confusão entre classes, indicando áreas onde o modelo pode não estar performando tão bem quanto o esperado.

Considerando essas métricas em conjunto, concluímos que o modelo não atingiu um desempenho satisfatório e há espaço para melhorias significativas. É importante continuar monitorando e refinando o modelo conforme necessário para garantir um desempenho ótimo em cenários de aplicação real, especialmente se utilizado em áreas mais críticas, como medicina, segurança ou finanças. Nessas áreas, pequenas imprecisões ou erros podem ter consequências significativas. A melhoria contínua do modelo garante sua eficácia e utilidade em aplicações do mundo real, onde a precisão e a confiabilidade são de extrema importância.

Como trabalhos futuros, pretende-se explorar alternativas para mitigar o *overfitting*, como o uso de técnicas de regularização e *data augmentation* mais avançado. Para lidar com a falta de armazenamento de memória sofrida durante o aumento de dados neste trabalho, pode-se explorar o uso de recursos como a arquitetura CUDA para processamento paralelo em GPUs. Além disso, investigar a coleta de dados mais diversificados e representativos e aprimorar a qualidade do pré-processamento dos dados podem contribuir para um desempenho aprimorado do modelo. Ainda, a delimitação por pontos do polígono em vez de *bounding boxes* pode ser uma abordagem interessante visando aprofundar a segmentação de objetos e a precisão das classificações. Por fim, a utilização de um modelo pré-treinado com *fine-tuning* poderia acelerar o processo de treinamento e melhorar o desempenho do modelo. Essas sugestões representam pontos promissores que foram levantados durante o desenvolvimento do modelo com a intenção de obter um desempenho superior e uma maior aplicabilidade em cenários do mundo real.

REFERÊNCIAS

[1] FERREIRA, M. E. et al. *Uso de imagens digitais na avaliação da cobertura do solo*. [S.l.]: Embrapa Cerrados Planaltina, 2001.

[2] AGRAWAL BRANDON AMOS, S. B. S. B. S. D. J. Z. K. A. *Differentiable Convex Optimization Layers*. 2019. <https://locuslab.github.io/2019-10-28-cvxpylayers/>. Accessed: 2024-04-15.

[3] SOUZA, V. S. S. Introdução à interpretabilidade de redes neurais convolucionais. 2023.

[4] LUCENA, O. A.; VELOSO, L. R.; LOPES, W. T. Implementaçao de um sistema de reconhecimento de objetos em imagens. *Revista de Tecnologia da Informação e Comunicação*, v. 6, n. 2, p. 34–42, 2016.

[5] LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015.

[6] RAWAT, W.; WANG, Z. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, MIT Press, v. 29, n. 9, p. 2352–2449, 2017.

[7] KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, AcM New York, NY, USA, v. 60, n. 6, p. 84–90, 2017.

[8] HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778.

[9] CHOLLET, F. *Deep learning with Python*. [S.l.]: Simon and Schuster, 2021.

[10] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press, 2016.

[11] ALBUQUERQUE, M. P. de; ALBUQUERQUE, M. P. de. Processamento de imagens: métodos e análises. *Centro Brasileiro de Pesquisas Físicas MCT*, 2000.

[12] CROSTA, A. P. *Processamento digital de imagens de sensoriamento remoto*. [S.l.]: UNICAMP/Instituto de Geociências, 1999.

[13] GONZALEZ, R. C.; WOODS, R. E. *Processamento de imagens digitais*. [S.l.]: Editora Blucher, 2000.

[14] QUEIROZ, J. E. R. de; GOMES, H. M. Introdução ao processamento digital de imagens. *Rita*, v. 13, n. 2, p. 11–42, 2006.

[15] THAI, H.-T. Machine learning for structural engineering: A state-of-the-art review. In: ELSEVIER. *Structures*. [S.l.], 2022. v. 38, p. 448–491.

[16] NASTESKI, V. An overview of the supervised machine learning methods. *Horizons. b*, v. 4, p. 51–62, 2017.

[17] MAHESH, B. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, v. 9, n. 1, p. 381–386, 2020.

[18] SILVA, T. L.; JR, M. M. G. Aprendizagem por reforço clássica e conexionista: análise de aplicações. *Anais do Encontro Anual de Tecnologia da Informação*, v. 5, n. 1, p. 299–299, 2015.

[19] HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2001.

[20] BRAGA, A. de P.; FERREIRA, A. C. P. de L.; LUDERMIR, T. B. *Redes neurais artificiais: teoria e aplicações*. [S.l.]: LTC editora, 2007.

[21] SPÖRL, C.; CASTRO, E.; LUCHIARI, A. Aplicação de redes neurais artificiais na construção de modelos de fragilidade ambiental. *Revista do Departamento de Geografia*, v. 21, p. 113–135, 2011.

[22] SANTOS, A. M. d. et al. Usando redes neurais artificiais e regressão logística na predição da hepatite a. *Revista Brasileira de Epidemiologia*, SciELO Public Health, v. 8, p. 117–126, 2005.

[23] HIROSE, Y.; YAMASHITA, K.; HIJIYA, S. Back-propagation algorithm which varies the number of hidden units. *Neural networks*, Elsevier, v. 4, n. 1, p. 61–66, 1991.

[24] ARAI, M. Bounds on the number of hidden units in binary-valued three-layer neural networks. *Neural Networks*, Elsevier, v. 6, n. 6, p. 855–860, 1993.

[25] FUJITA, O. Statistical estimation of the number of hidden units for feedforward neural networks. *Neural networks*, Elsevier, v. 11, n. 5, p. 851–859, 1998.

[26] SILVA, I. N. d.; SPATTI, D. H.; FLAUZINO, R. A. Redes neurais artificiais para engenharia e ciências aplicadas. 2010.

[27] GHARAT. *What, Why and Which? Activation Functions*. 2019. <https://medium.com/@snaily16/what-why-and-which-activation-functions-b2bf748c0441>. Accessed: 2024-04-01.

[28] NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. [S.l.: s.n.], 2010. p. 807–814.

[29] LI, X. et al. Understanding the disharmony between dropout and batch normalization by variance shift. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. [S.l.: s.n.], 2019. p. 2682–2690.

[30] MAAS, A. L. et al. Rectifier nonlinearities improve neural network acoustic models. In: ATLANTA, GA. *Proc. icml*. [S.l.], 2013. v. 30, n. 1, p. 3.

[31] HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 1026–1034.

[32] BARBER, D.; SAAD, D.; SOLLICH, P. Finite-size effects in on-line learning of multilayer neural networks. *Europhysics letters*, IOP Publishing, v. 34, n. 2, p. 151, 1996.

[33] FERREIRA, A. d. S. Redes neurais convolucionais profundas na detecçao de plantas daninhas em lavoura de soja. 2017.

[34] ZHANG, Q.; WU, Y. N.; ZHU, S.-C. Interpretable convolutional neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2018. p. 8827–8836.

[35] GHOLAMALINEZHAD, H.; KHOSRAVI, H. Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*, 2020.

[36] MASÍS, S. *Interpretable Machine Learning with Python: Build explainable, fair, and robust high-performance models with hands-on, real-world examples*. [S.l.]: Packt Publishing Ltd, 2023.

[37] RIESENHUBER, M.; POGGIO, T. Hierarchical models of object recognition in cortex. *Nature neuroscience*, Nature Publishing Group, v. 2, n. 11, p. 1019–1025, 1999.

[38] LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Ieee, v. 86, n. 11, p. 2278–2324, 1998.

[39] PINTO, A. H. M. *Um sistema de reconhecimento de objetos incorporado a um robô humanoide aplicado na educação*. Tese (Doutorado) — Universidade de São Paulo, 2016.

[40] MURTY, M. N.; DEVI, V. S. *Introduction to pattern recognition and machine learning*. [S.l.]: World Scientific, 2015. v. 5.

[41] JOGIN, M. et al. Feature extraction using convolution neural networks (cnn) and deep learning. In: IEEE. *2018 3rd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT)*. [S.l.], 2018. p. 2319–2323.

[42] INDIRA, D. et al. A review on fruit recognition and feature evaluation using cnn. *Materials Today: Proceedings*, Elsevier, v. 80, p. 3438–3443, 2023.

[43] SANTOS, V. N. d. *Reconhecimento de objetos em uma cena utilizando redes neurais convolucionais*. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2018.

[44] KADHIM, M. R.; OLEIWI, B. K. Blind assistive system based on real time object recognition using machine learning. *Engineering and Technology Journal*, University of Technology-Iraq, v. 40, n. 1, p. 159–165, 2022.

[45] UDGIRKAR, S. et al. Object detection system for blind people. *International Journal of Innovative Research in Computer and Communication Engineering*, v. 4, n. 9, 2016.

[46] COTRIM, W. d. S. Inteligência artificial aplicada a modelagem de processos da indústria de alimentos. Universidade Federal de Viçosa, 2021.

[47] LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: SPRINGER. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. [S.l.], 2014. p. 740–755.

[48] SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understanding machine learning: From theory to algorithms*. [S.l.]: Cambridge university press, 2014.

[49] LIANG, J. Confusion matrix: Machine learning. *POGIL Activity Clearinghouse*, v. 3, n. 4, 2022.

[50] GÉRON, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. [S.l.]: "O'Reilly Media, Inc.", 2022.

[51] HE, H.; GARCIA, E. A. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, Ieee, v. 21, n. 9, p. 1263–1284, 2009.

[52] HACKELING, G. *Mastering Machine Learning with scikit-learn*. [S.l.]: Packt Publishing Ltd, 2017.

[53] SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on image data augmentation for deep learning. *Journal of big data*, Springer, v. 6, n. 1, p. 1–48, 2019.

[54] GONZALEZ, R. C. *Digital image processing*. [S.l.]: Pearson education india, 2009.

[55] WALT, S. V. D.; COLBERT, S. C.; VAROQUAUX, G. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, IEEE, v. 13, n. 2, p. 22–30, 2011.

[56] HARRIS, C. R. et al. Array programming with numpy. *Nature*, Nature Publishing Group UK London, v. 585, n. 7825, p. 357–362, 2020.

[57] FINLAYSON, G. D.; SCHIELE, B.; CROWLEY, J. L. Comprehensive colour image normalization. In: SPRINGER. *Computer Vision—ECCV'98: 5th European Conference on Computer Vision Freiburg, Germany, June, 2–6, 1998 Proceedings, Volume I 5*. [S.l.], 1998. p. 475–490.

[58] GULLI, A.; PAL, S. *Deep learning with Keras*. [S.l.]: Packt Publishing Ltd, 2017.

[59] ABADI, M. et al. {TensorFlow}: a system for {Large-Scale} machine learning. In: *12th USENIX symposium on operating systems design and implementation (OSDI 16)*. [S.l.: s.n.], 2016. p. 265–283.

[60] SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.

[61] MASTERS, D.; LUSCHI, C. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.