INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMAZONAS
CAMPUS MANAUS DISTRITO INDUSTRIAL
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

KAÍNY MEDEIROS MACIEL

USO DE REDES NEURAIS PARA CLASSIFICAÇÃO DE RESÍDUOS SÓLIDOS

MANAUS – AM
2024

KAÍNY MEDEIROS MACIEL

USO DE REDES NEURAIS PARA CLASSIFICAÇÃO DE RESÍDUOS SÓLIDOS

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, Campus Manaus Distrito Industrial, como requisito para obtenção do titulo de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Alyson de Jesus dos Santos

MANAUS – AM
2024

Dados Internacionais de Catalogação na Publicação (CIP)

M152u

Maciel, Kaíny Medeiros.
Uso de redes neurais para classificação de resíduos sólidos / Kaíny Medeiros Maciel. — Manaus, 2024.
79f.: il. color.

Monografia (Graduação) — Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, *Campus* Manaus Distrito Industrial, Curso de Engenharia de Controle e Automação, 2024.
Orientador: Prof.º Alyson de Jesus dos Santos, Dr.

1. Redes Neurais. 2. Visão computacional. 3. Imagens. I. Santos, Alyson de Jesus dos.. II. Instituto Federal de Educação, Ciência e Tecnologia do Amazonas. III. Título.

CDD 629.89

Elaborada por Oziane Romualdo de Souza (CRB11/ nº 734)

ANEXO 7

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

Aos 26 dias do mês de Novembro, de 2024, às 13 h, o(a) discente Kainy Medeiros Maciel apresentou o seu Trabalho de Conclusão de Curso para avaliação da Banca Examinadora constituída pelos seguintes integrantes: Prof(a). Dr. Alyson de Jesus dos Santos (docente-orientador), Prof(a). Msc. Marlos André Silva Rodrigues (co-orientador), Prof(a). Msc. José Fábio de Lima Nascimento (Membro 1) e Prof(a). Msc. Sérgio Costa Martins de Alencar (Membro 2). A sessão pública de defesa foi aberta pelo(a) presidente da banca, que apresentou a Banca Examinadora e deu continuidade aos trabalhos, fazendo uma breve referência ao TCC, que tem como título Uso de Redes Neurais Para a Classificação de Resíduos Sólidos. Na sequência, o(a) discente teve até 30 minutos para a comunicação oral de seu trabalho. Cada integrante da banca examinadora fez suas arguições após a defesa do mesmo. Ouvidas as explicações do(a) discente, a banca examinadora, reunida em caráter sigiloso, para proceder à avaliação final, deliberou e decidiu pela APROVAÇÃO com média final 9,0 (Nove) do referido trabalho.

Foi dada ciência ao(à) discente que a versão final do trabalho deverá ser entregue até o dia 26/12/2024, com as devidas alterações sugeridas pela banca. Nada mais havendo a tratar, a sessão foi encerrada às 14h 00 min, sendo lavrada a presente ata, que, uma vez aprovada, foi assinada por todos os membros da Banca Examinadora e pelo(a) discente.

Prof.(a) Orientador(a)/Presidente: Alyson de Jesus dos Santos

Prof.(a) Co-Orientador(a):

Prof.(a) Avaliador 1:

Prof.(a) Avaliador 2:

Discente: Kainy Medeiros Maciel

AGRADECIMENTOS

A graduação é uma daquelas jornadas que você começa achando que será desafiadora, mas só descobre a real definição de “desafio” quando já está imerso em provas e projetos. Foi um percurso importante que não apenas moldou meu conhecimento, mas também meu caráter – e eu não teria chegado até aqui sem as pessoas incríveis que estiveram ao meu lado.

Primeiramente, agradeço aos meus pais e à minha irmã que, com amor e paciência, me ensinaram o valor da persistência e da dedicação. Como professores, eles me mostraram não apenas a importância do conhecimento, mas também que as maiores conquistas vêm com esforço e trabalho duro. O incentivo deles foi essencial para que eu concluísse essa caminhada e continuasse sonhando com novos passos na vida acadêmica.

Aos amigos que, mesmo distantes, estiveram comigo em diversos momentos, arrancando risadas e me convencendo a "esfriar a cabeça" jogando LoLzinho. Eles me ajudaram a encontrar equilíbrio nas horas difíceis e tornar o processo mais leve.

Por último, agradeço a todos que carregaram minha ansiedade e me lembraram de que um trabalho acadêmico pode até ser difícil, mas nunca impossível. Obrigada pelo apoio que nenhum livro poderia me oferecer!

"Tudo parece impossível até que seja feito."

(Nelson Mandela)

RESUMO:

O objetivo deste trabalho é desenvolver um sistema automatizado para a classificação de resíduos sólidos utilizando redes neurais convolucionais. Considerando os problemas na gestão de resíduos e a necessidade de práticas mais sustentáveis, a proposta combina aprendizado de máquina e processamento de imagens para aprimorar a coleta seletiva. O uso de redes neurais se apresenta como uma solução eficiente e inovadora, superando limitações de métodos convencionais. Entretanto, é válido considerar a necessidade de conjuntos de dados diversificados, frente a baixa variabilidade das imagens nessa área. A pesquisa foi estruturada em capítulos que abordam a revisão bibliográfica, a definição de um banco de dados de imagens, o treinamento da rede neural e a avaliação da precisão do sistema. Espera-se que os resultados contribuam para uma separação mais eficaz de resíduos, além de promover sustentabilidade e maior conscientização sobre a reciclagem.

Palavras-chave: Visão computacional, Redes neurais, Classificação de imagens, Resíduos sólidos.

ABSTRACT:

The objective of this work is to develop an automated system for the classification of solid waste using convolutional neural networks. Considering the challenges in waste management and the need for more sustainable practices, the proposed approach combines machine learning and image processing to enhance selective collection. The use of neural networks presents itself as an efficient and innovative solution, overcoming the limitations of conventional methods. However, it is important to consider the need for diversified datasets due to the low variability of images in this field. The research is structured into chapters that cover the literature review, the definition of an image database, the training of the neural network, and the evaluation of the system's accuracy. The results are expected to contribute to more effective waste separation, while also promoting sustainability and raising awareness about recycling.

Keywords: Computer vision, Neural networks, Image classification, Solid waste.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABREMA - Associação Brasileira de Resíduos e Meio Ambiente

CNN – *Convolucional Neural Network*

GEE - Gases de Efeito Estufa

GWMO 2024 - *Global Waste Management Outlook 2024*

IA - Inteligência Artificial

ISWA - *International Solid Waste Association*

MDR - Ministério do Desenvolvimento Regional

MMA - Ministério do Meio Ambiente

PLANARES - Plano Nacional de Resíduos Sólidos

PLANSAB - Plano Nacional de Saneamento Básico

PNRS - Política Nacional de Resíduos Sólidos

RSU - Resíduo Sólido Urbano

SINIR - Sistema Nacional de Informações sobre a Gestão dos Resíduos Sólidos

SNS - Secretaria Nacional de Saneamento

SQA - Secretaria de Qualidade Ambiental

TCC - Trabalho de Conclusão de Curso

UNEP - *United Nations Environment Programme*

SUMÁRIO

1 INTRODUÇÃO

Com o avanço da sociedade, em termos populacionais e tecnológicos, a gestão adequada dos resíduos torna-se cada vez mais indispensável em razão do aumento da produção de lixo e à escassez de sistemas eficientes de coleta seletiva, os quais são essenciais para a separação e destinação apropriada dos resíduos.

De acordo com o Panorama dos Resíduos Sólidos no Brasil 2023, levantamento publicado pela Associação Brasileira de Resíduos e Meio Ambiente (ABREMA), o Brasil ainda está em fase inicial de implementação da coleta seletiva de resíduos sólidos. Considerando todos os municípios do país, a média da população urbana atendida alcança apenas 14,7% dos habitantes, contrastando com a região Norte que atende somente 2,0% da população urbana com coleta seletiva (ABREMA, 2023).

A reciclagem desempenha um papel importante no desenvolvimento de uma sociedade sustentável, pois substitui a extração de matérias-primas pela reutilização de resíduos considerados desnecessários e reduz o acúmulo de lixo. Contudo, as técnicas convencionais de reciclagem geralmente são manuais e consideradas demoradas e ineficientes, além de serem realizadas em condições de trabalho insalubres devido à contaminação dos resíduos.

Neste cenário, o aproveitamento adequado dos resíduos e a minimização do desperdício são fundamentais para mitigar os impactos socioambientais e promover a sustentabilidade. Uma solução promissora seria utilizar a visão computacional para identificar e classificar diferentes tipos de resíduos com mais precisão e eficiência do que os métodos tradicionais.

1.1 JUSTIFICATIVA

O crescimento populacional, a industrialização e o consumismo aumentam substancialmente a geração de resíduo sólido urbano (RSU). Segundo ABREMA (2023, p. 20): “Estima-se que o brasileiro tenha gerado uma média de 1,04 kg de RSU por dia em 2022.”, logo, as concessionárias de limpeza urbana precisam recolher mais lixo e, com essa demanda, ocorre a superlotação dos aterros sanitários.

Embora seja a destinação mais comum, os aterros sanitários têm uma vida útil limitada, geralmente em torno de 10 anos, mas nem todos alcançam esse prazo. Quando o aterro esgota sua capacidade, é preciso fechá-lo e implementar medidas adequadas para diminuir os impactos ambientais. Contudo, mesmo após o encerramento das operações, gases e chorume continuam sendo liberados por pelo menos 15 anos.

Com a previsão para escassez dos aterros sanitários, faz-se necessária a busca por novas formas de lidar com o RSU, principalmente as sustentáveis, mas esta alternativa também encarece para o empresário, pois requer mais trabalho e injeção de recursos por parte das concessionárias, incluindo novos ambientes de trabalho, trabalhadores, treinamentos etc.

A solução mais viável seria aderir ao descarte consciente de lixo, que diminui as despesas com limpeza pública e contribui com o meio ambiente. Porém, o ser humano não é perfeito e, muitas vezes, tem dificuldade em mudar certos hábitos, além de que o investimento em reeducação seria exorbitante para que pudessem surtir efeitos.

O sistema a ser desenvolvido neste trabalho se justifica pois auxiliará o cidadão ao gerar sustentabilidade e comodidade, o empresário ao automatizar e agilizar o processo e diminuir os custos a longo prazo, e o país ao introduzir e avançar a coleta seletiva de maneira mais prática.

1.2 OBJETIVO

O objetivo principal deste Trabalho de Conclusão de Curso (TCC) consiste em desenvolver um sistema inteligente, baseado em visão computacional, que aprimore o processo de separação de resíduos sólidos urbanos. Para isso, serão utilizados algoritmos de aprendizado de máquina, que têm capacidade de interpretar imagens de maneira mais precisa e rápida quando comparados aos métodos convencionais.

Consequentemente, o presente estudo tem como proposta analisar a eficiência do uso de redes neurais na classificação de resíduos sólidos, com o intuito de contribuir para a evolução das pesquisas relacionadas a esse tema.

1.3 OBJETIVOS ESPECÍFICOS

Para a obtenção de resultados satisfatórios, foram determinados os seguintes objetivos específicos:

a) Estudar bibliografias e tecnologias que possam auxiliar no desenvolvimento do projeto;
b) Escolher a técnica de construção da rede neural mais indicada considerando o objetivo geral;
c) Treinar a rede neural através do banco de dados escolhido para a classificação dos resíduos;
d) Testar o sistema avaliando sua precisão em determinar o tipo de material;
e) Analisar os resultados obtidos e identificar as capacidades e limitações do sistema;

1.4 METODOLOGIA

Este estudo aplicará a metodologia de uma pesquisa aplicada, iniciando com uma revisão bibliográfica para compreender os fundamentos teóricos e os avanços recentes na área de visão computacional, explorando livros, dissertações, artigos e trabalhos acadêmicos. Em seguida, serão definidos os métodos de obtenção das imagens de resíduos sólidos e treinamento da rede neural.

O desenvolvimento do sistema de coleta seletiva será realizado por meio da aplicação de redes neurais e aprendizado de máquina, visando a classificação e separação eficaz dos resíduos. Então, serão realizados testes para verificar a precisão e a eficácia do sistema, incluindo sua aplicação em diferentes contextos e comparação com métodos tradicionais de coleta seletiva.

Com base nos resultados dos testes, será necessário realizar uma análise detalhada e interpretar os dados obtidos para verificar se os objetivos propostos foram atingidos, se as hipóteses levantadas foram confirmadas, bem como avaliar se os problemas práticos inicialmente identificados foram devidamente solucionados e se os resultados são consistentes com as expectativas do estudo.

1.5 ESTRUTURA DO TCC

O restante do trabalho seguirá a seguinte estrutura: no capítulo 2 serão expostos os fundamentos teóricos essenciais para o desenvolvimento do sistema, incluindo conceitos sobre inteligência artificial, visão computacional e técnicas de classificação de imagens, bem como a linguagem de programação, bibliotecas e ferramentas de software empregadas. O capítulo 3 discorrerá sobre trabalhos relacionados já realizados, discutindo seus métodos e resultados. No capítulo 4, serão detalhados os materiais e métodos adotados para o desenvolvimento e aplicação do sistema, abrangendo o processamento das imagens e a construção e treinamento da rede neural. O capítulo 5 apresentará os resultados e discussões, visando avaliar a eficácia do trabalho proposto. Por fim, o capítulo 6 trará a conclusão do estudo, completando a ideia inicialmente proposta.

2 REFERENCIAL TEÓRICO

Este capítulo abordará a situação dos resíduos sólidos no Brasil, a coleta seletiva e os meios de descarte utilizados. Nesse contexto, também serão apresentados os fundamentos conceituais técnicos necessários para o entendimento do projeto.

2.1 RESÍDUOS SÓLIDOS URBANOS

Além da definição padrão de RSU adotada pelo Sistema Nacional de Informações sobre a Gestão dos Resíduos Sólidos (SINIR), a Lei do Novo Marco Legal do Saneamento determinou que "[...] os resíduos originários de atividades comerciais, industriais e de serviços cuja responsabilidade pelo manejo não seja atribuída ao gerador pode, por decisão do poder público, ser considerado resíduo sólido urbano." (BRASIL, 2020, p. 1).

> Resíduos Sólidos Urbanos (RSU) são aqueles originários de atividades domésticas em residências urbanas (resíduos domiciliares) e os originários da varrição, limpeza de logradouros e vias públicas e outros serviços de limpeza urbana (resíduos de limpeza urbana) (BRASIL, 2020, p. 1).

Embora este trabalho contemple prioritariamente os resíduos originários de atividades urbanas, vale ressaltar que, de acordo com a Política Nacional de Resíduos Sólidos (PNRS), há um conjunto de tipologias de resíduos que os classifica quanto à sua origem e periculosidade:

a) **Quanto à origem:**
   - resíduos domiciliares;
   - resíduos de limpeza urbana;
   - resíduos sólidos urbanos;
   - resíduos de estabelecimentos comerciais e prestadores de serviços;
   - resíduos dos serviços públicos de saneamento básico;
   - resíduos industriais;
   - resíduos de serviços de saúde;
   - resíduos da construção civil;
   - resíduos agrossilvopastoris;
   - resíduos de serviços de transportes:;
   - resíduos de mineração;

b) **Quanto à periculosidade:**
   - resíduos perigosos;
   - resíduos não perigosos.

Independentemente de sua classificação, todo resíduo sólido deve ser gerenciado seguindo o ciclo mostrado na figura abaixo.

FIGURA 1 – Etapas de gerenciamento do RSU

Fonte: BRASIL (2020a)

O ciclo de gerenciamento dos resíduos sólidos urbanos começa com sua geração, que pode ocorrer em locais como residências, estabelecimentos comerciais, rurais, industriais e de serviços. Geralmente, os RSU são recolhidos por equipes de coleta, mas dependendo da infraestrutura e das políticas locais de gestão de resíduos, a coleta pode ser feita de maneiras diferentes, através de contêineres comunitários ou pontos de entrega voluntária.

Após a coleta, os RSU são transportados para locais de transbordo, onde podem ser compactados e transferidos para veículos maiores. Durante o processo, esses RSU são submetidos à triagem considerando sua disposição final, podendo ser reuso, reciclagem ou tratamento.

Em seguida, os resíduos selecionados são destinados para usinas de reciclagem ou instalações de compostagem. Se, depois de esgotadas todas as possibilidades de tratamento e recuperação, ainda sobrarem resíduos, estes passarão a ser rejeitos e não apresentarão outra possibilidade que não a disposição final ambientalmente adequada, como aterros sanitários ou incineração.

2.1.1 GERAÇÃO E DESCARTE

O crescimento acelerado e desordenado das cidades brasileiras, em conjunto com o aumento da população e o consumo em grande escala de produtos industrializados e descartáveis, tem causado um aumento expressivo na quantidade de RSU (BRASIL, 2019).

FIGURA 2 – Geração de RSU no Brasil em 2022

Fonte: ABREMA (2023)

A ABREMA considerou o Censo Demográfico 2022 para estimar que cerca de 77,1 milhões de toneladas de RSU foram produzidas no país. Isso equivale a mais de 211 mil toneladas de resíduos geradas diariamente, ou aproximadamente 380 kg por habitante ao ano (ABREMA, 2023). As projeções indicam que a produção de resíduos no Brasil deverá aumentar em mais de 50% até 2050, alcançando 120 milhões de toneladas por ano.

Segundo a Associação Internacional de Resíduos Sólidos (2021, p. 9): "Estima-se que, no cenário vigente de produção de bens de consumo, a geração de resíduos sólidos urbanos aumentará em todo o mundo, passando de 2 bilhões de toneladas/ano em 2016 para 3,4 bilhões de toneladas em 2050 [...]".

> A *International Solid Waste Association* – ISWA é uma associação internacional, não governamental e sem fins lucrativos, que atua pelo interesse público de promover e desenvolver o setor de resíduos sólidos ao redor do mundo para uma sociedade sustentável, da qual a ABRELPE é Representante no Brasil (ABRELPE, 2023, p. 1).

A ISWA e o *United Nations Environment Programme* – UNEP publicaram o relatório *Global Waste Management Outlook* 2024 - GWMO 2024, que oferece uma avaliação atualizada do gerenciamento global de resíduos e uma análise dos dados relacionados ao manejo de resíduos sólidos municipais em todo o mundo.

Segundo o estudo, o Brasil lidera como o maior gerador de resíduos urbanos na América Latina e no Caribe, dentre esses, cerca de 40% são descartados de forma inapropriada, como em lixões e queima a céu aberto.

A maioria dos resíduos descartados, seja nas vias públicas, rios, terrenos baldios ou através da queima a céu aberto, por não serem coletados pelo serviço público de limpeza urbana, acabam sendo transportados pelas águas pluviais até os rios e, eventualmente, para o mar (BRASIL, 2022b).

> Práticas indiscriminadas de descarte de resíduos podem introduzir produtos químicos perigosos no solo, corpos d'água e no ar, causando danos a longo prazo, potencialmente irreversíveis, à flora e fauna locais, impactando negativamente a biodiversidade, prejudicando ecossistemas inteiros e entrando na cadeia alimentar humana (UNEP et al, 2024, p. 12, tradução nossa).

O descarte inadequado não prejudica apenas a saúde pública, a economia e o meio ambiente, causando danos aos ecossistemas terrestres e aquáticos, mas também dificulta a mensuração precisa da quantidade de resíduos gerados. Entretando, para compreender o panorama brasileiro nesse aspecto, é crucial identificar quais resíduos são gerados, em que volume e em quais locais. Assim, devido aos fatores já citados e à ausência de procedimentos sistematizados para coletar os dados de maneira precisa, o volume total de resíduos gerados é, geralmente, estimado com base em métricas e critérios pré-estabelecidos.

> Em resumo, em um futuro próximo, veremos um aumento drástico na geração de resíduos sólidos urbanos em todo o mundo, o que exigirá um aumento das capacidades de coleta e tratamento combinadas com aplicações úteis para os materiais recuperados. Isso exigirá um aumento significativo de recursos financeiros que, em paralelo, exigirá medidas para garantir que os fundos arrecadados sejam realmente utilizados para essa finalidade, com forte governança e medidas de transparência (ISWA, 2021, p. 9).

O relatório afirma que em 2020, o custo direto global do gerenciamento de resíduos foi estimado em US$ 252 bilhões. Considerando os custos ocultos relacionados à poluição, saúde precária e mudanças climáticas decorrentes de práticas inadequadas de descarte de resíduos, o custo aumenta para US$ 361 bilhões. Sem a implementação urgente de medidas para o gerenciamento adequado de resíduos, esse custo global anual poderá quase dobrar até 2050, alcançando impressionantes US$ 640,3 bilhões (PNUMA, 2024).

2.1.2 COLETA

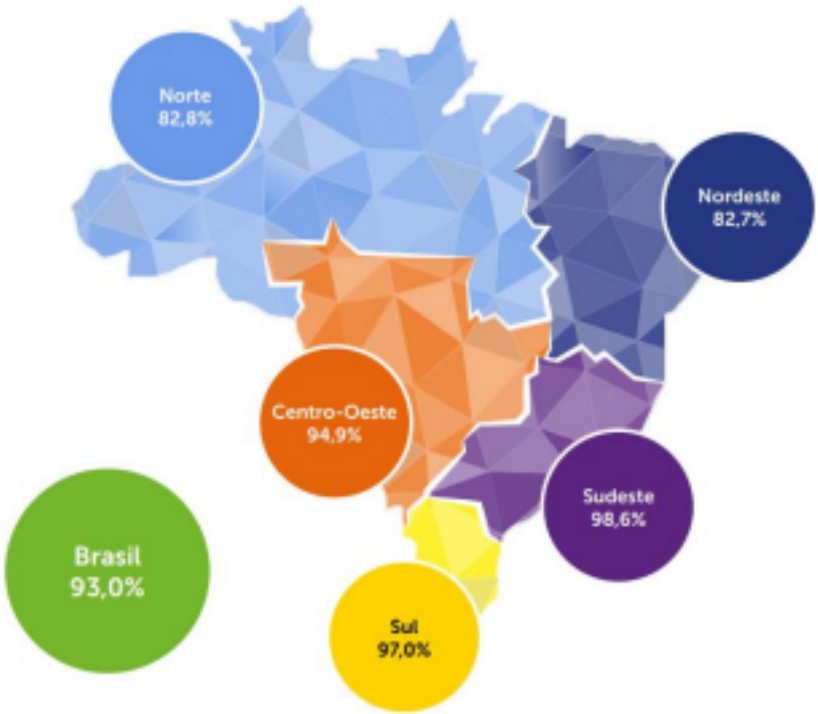
A coleta é uma etapa fundamental no gerenciamento de RSU, na qual equipes compostas por trabalhadores, geralmente especializados e equipados com veículos apropriados, percorrem as áreas designadas para recolher esses resíduos. Entretanto, devido à insuficiência do serviço público de coleta, aliada à falta de consciência sanitária e ambiental da população, nem todos os resíduos gerados são coletados de maneira adequada (BRASIL, 2022b).

FIGURA 3 – Coleta de RSU no Brasil em 2022

Fonte: ABREMA (2023)

No ano de 2022, aproximadamente 93% dos resíduos gerados no Brasil foram adequadamente recolhidos, totalizando mais de 196 mil toneladas de RSU coletadas diariamente. Embora esse número represente uma porcentagem significativa, é essencial ressaltar que os 7% de resíduos não coletados equivalem a mais de 5 milhões de toneladas descartadas de maneira inadequada. Os impactos gerados pelos resíduos não coletados reforçam a importância de melhorar os sistemas de coleta e gestão de resíduos (ABREMA, 2023).

FIGURA 4 – Percentual de RSU coletados por região em 2022



Fonte: BRASIL (2019)

Os percentuais regionais demonstram que as regiões Sul, Sudeste e Centro-Oeste superam a média nacional em termos de coleta, enquanto o Norte e o Nordeste coletam cerca de 83% dos resíduos gerados, destacando as discrepâncias na gestão regional de RSU no Brasil.

Segundo o GWMO divulgado em 2024, estima-se que aproximadamente "2,7 bilhões de pessoas não têm seu lixo coletado" (UNEP et al, 2024, p. 23, tradução nossa). Essa realidade é mostrada no Brasil, onde uma em onze pessoas não possui acesso aos serviços básicos de limpeza urbana, resultando em mais de 5 milhões de toneladas de resíduos sólidos não coletadas anualmente.

2.1.2.1 COLETA CONVENCIONAL

A coleta convencional, também chamada de coleta indiferenciada, consiste em recolher os resíduos sólidos urbanos sem qualquer separação prévia. Geralmente, envolve um sistema periódico de recolhimento, onde caminhões compactadores percorrem rotas predefinidas para coletar os resíduos diretamente das residências, comércios e pontos de coleta pública.

Os resíduos, sejam eles orgânicos, recicláveis ou não recicláveis, são coletados conjuntamente e acondicionados com rejeitos e outros resíduos, produzindo um resíduo misto, com maior grau de impurezas. Esse fator gera contaminação entre diferentes tipos de materiais e limita a gama de aplicações do composto produzido. Por exemplo, resíduos orgânicos podem sujar papéis, plásticos e metais, tornando-os menos viáveis para a reciclagem.

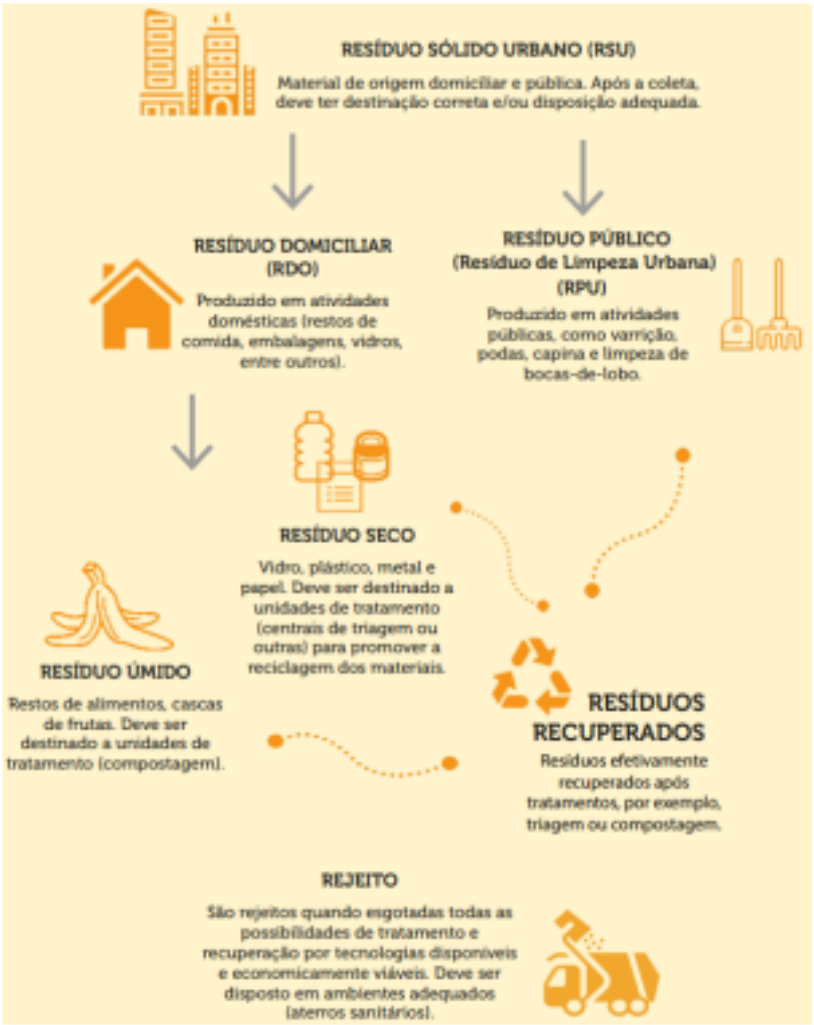
A presença de resíduo orgânico em lixões causa riscos à saúde pública e degradação ambiental, pois atrai animais vetoriais e portadores de doenças; polui o solo e o lençol freático com chorume, que contém alta carga orgânica e nitrogenada; e produz biogás com altos níveis de gás metano, que retêm calor na atmosfera e contribui para o efeito estufa. Os resíduos orgânicos também reduzem a vida útil e aumentam os custos operacionais e de manutenção dos aterros sanitários, pois é necessária a implementação de métodos e tecnologias voltados à coleta e tratamento do chorume e dos gases gerados pela decomposição da matéria orgânica, principalmente do metano.

Embora a coleta convencional apresente diversas desvantagens, continua sendo amplamente utilizada no Brasil devido à sua simplicidade operacional e à menor exigência de conscientização e colaboração da população.

2.1.2.2 COLETA SELETIVA

A coleta seletiva, ou diferenciada, é "definida como a coleta dos resíduos sólidos previamente separados, de acordo com a sua constituição ou composição [...]" (BRASIL, 2022b, p. 23). Essa separação é realizada pelo próprio gerador do resíduo no local onde ele foi produzido.

FIGURA 5 – Ciclo de coleta seletiva de RSU



Fonte: BRASIL (2022a)

Para uma disposição adequada dos resíduos, é essencial separá-los em pelo menos três frações distintas: resíduos orgânicos, resíduos recicláveis secos e resíduos não recicláveis, conhecidos como rejeitos. A segregação prévia dos resíduos orgânicos é particularmente importante, pois possibilita a produção de um composto de melhor qualidade para utilização no solo e na agricultura, reduzindo a contaminação por metais pesados, compostos tóxicos, microplásticos e outras substâncias prejudiciais. A não utilização de sacolas ou o uso de sacolas compostáveis para acondicionar os resíduos também melhora a qualidade do composto final (BRASIL, 2019).

Após o devido condicionamento, os resíduos são coletados pelo serviço municipal ou entregues em pontos específicos de coleta. É importante destacar que a coleta é considerada efetivamente 'seletiva' apenas quando o prestador de serviço realiza a coleta de forma diferenciada. Além disso, nos sistemas de coleta de resíduos, ainda há grande volume de resíduos misturados, destacando a necessidade de ações de educação ambiental direcionadas à população.

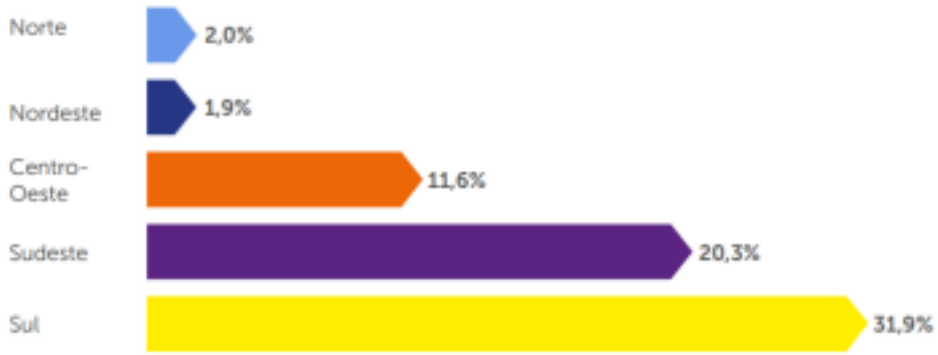
No Brasil, a implementação da coleta seletiva porta a porta ainda está em fase inicial, atingindo 69,7 milhões de habitantes. No entanto, considerando a média da população urbana atendida, por município, a coleta seletiva porta a porta alcança apenas 14,7% dos habitantes.

FIGURA 6 – População atendida com coleta seletiva porta a porta

Fonte: ABREMA (2023)

Os municípios da região Sul apresentam a maior média de cobertura, atendendo a 31,9% da população urbana, em compensação, a região Nordeste atende somente 1,9% da população urbana (BRASIL, 2022a). No entanto, mesmo que o município afirme oferecer coleta seletiva, isso não garante que seu território seja atendido integralmente.

FIGURA 7 – População regional atendida com coleta seletiva porta a porta



Fonte: ABREMA (2023)

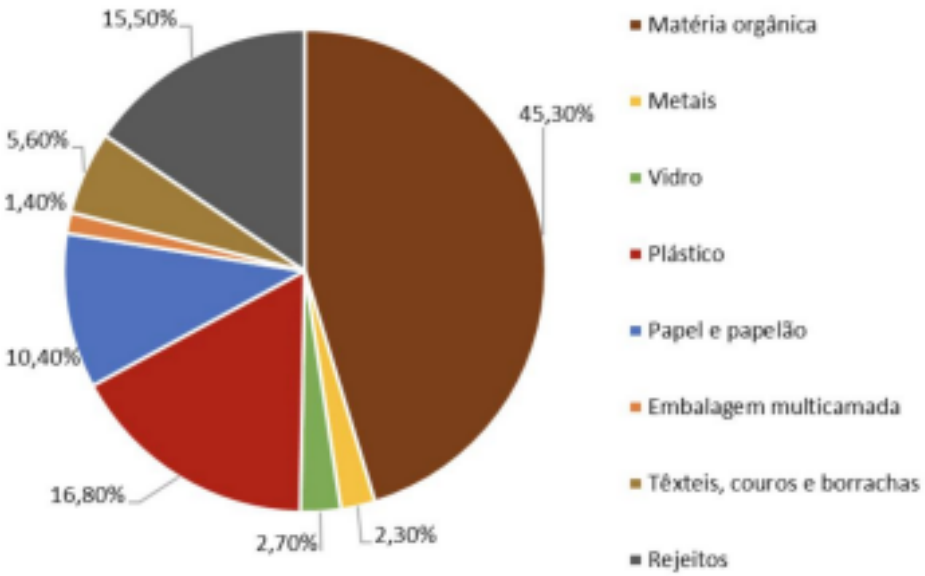
Embora os desafios sejam significativos, os benefícios ambientais, econômicos e sociais da coleta seletiva justificam os esforços para sua implementação e expansão. Com investimentos adequados, políticas públicas eficazes e a participação ativa da população, é possível aumentar as taxas de reciclagem, reduzir o impacto ambiental e promover um uso mais eficiente dos recursos no país.

2.1.3 COMPOSIÇÃO

O SINIR é o sistema responsável por receber, consolidar e divulgar dados e informações sobre a gestão de resíduos sólidos no Brasil, dentre elas a composição gravimétrica dos resíduos sólidos urbanos federais.

Conforme mostrado na figura abaixo, os resíduos orgânicos são a principal fração dos RSU coletados, totalizando 45,3%. Essa categoria inclui sobras de alimentos, resíduos verdes e madeiras, evidenciando sua contribuição significativa para o total de resíduos gerados e a importância de separá-los desde a fonte.

FIGURA 8 – Composição gravimétrica de RSU no Brasil



Fonte: BRASIL (2020b)

Os resíduos recicláveis secos compreendem 33,6% do total e são compostos principalmente por plásticos, papel, papelão, vidro, metais e embalagens multicamadas, que são frequentemente usados em produtos que requerem proteção contra fatores ambientais adversos, como umidade, luz e oxigênio. Por outro lado, os resíduos restantes constituem 21,1% do total e englobam resíduos têxteis, couros, borrachas e rejeitos, sendo predominantemente compostos por resíduos sanitários.

A diversidade na composição dos RSU reforça a complexidade da gestão de resíduos e a importância de estratégias eficazes para cada categoria de resíduo, além de promover práticas sustentáveis de manejo de resíduos no Brasil.

2.1.4 DESTINAÇÃO FINAL

A destinação final de resíduos sólidos urbanos é um conjunto de atividades e processos que ocorrem após a coleta dos resíduos, visando o tratamento, aproveitamento e reutilização dos materiais para minimizar a quantidade que precisa ser descartada de forma definitiva. O objetivo principal da destinação final é reduzir o volume de resíduos que chega à disposição final, promovendo a sustentabilidade e a conservação de recursos naturais.

> A PNRS, em seu art. 3º, inciso VII, definiu que destinação final ambientalmente adequada compreende a reutilização, a reciclagem, a compostagem, a recuperação e o aproveitamento energético ou outras destinações admitidas pelos órgãos competentes [...] (BRASIL, 2022a, p. 27).

A destinação adequada dos resíduos desempenha um papel crucial na redução das emissões de Gases de Efeito Estufa (GEE), utilizando tecnologias limpas e de baixas emissões. Isso ocorre através da conversão do metano em CO2, do uso de materiais secundários na indústria, de combustíveis derivados de resíduos no setor energético e de compostos na agricultura.

As diversas formas de destinação final ambientalmente adequadas são complementares e devem seguir a gradação legal, visando economizar recursos naturais, energia, recursos financeiros e promover a sustentabilidade econômica dos serviços. Com exceção da disposição final, todas as demais formas de destinação promovem um uso mais eficiente dos recursos naturais.

Contudo, o avanço na destinação final adequada de resíduos depende da implementação efetiva de políticas públicas, envolvendo governos, empresas e sociedade civil para integrar práticas sustentáveis e reduzir custos. É essencial considerar toda a cadeia de produção e consumo, desde a concepção de produtos que minimizem a geração de resíduos até a promoção da redução, reutilização, reciclagem e recuperação. Além disso, são necessárias iniciativas para incentivar o consumo e descarte conscientes, junto com o apoio financeiro e incentivos econômicos para tornar todas as etapas viáveis e atrativas.

2.1.5 DISPOSIÇÃO FINAL

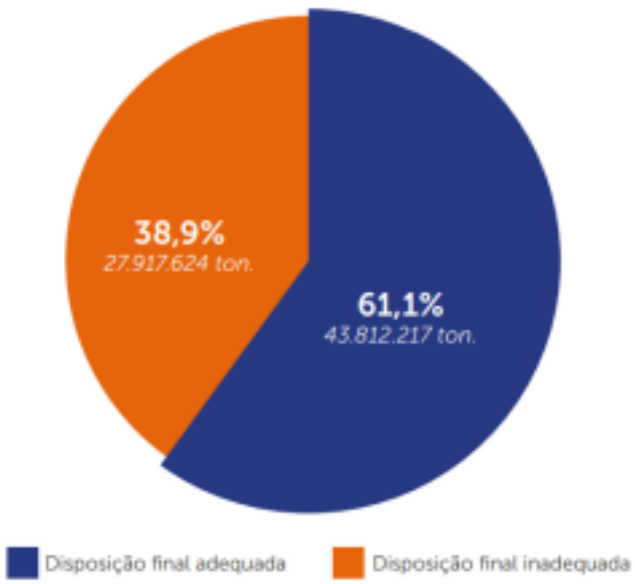
A disposição final de Resíduos Sólidos Urbanos é a última etapa no processo de gerenciamento de resíduos, onde os resíduos são permanentemente colocados em locais especialmente projetados e gerenciados para minimizar os impactos. Esta etapa é essencial para garantir que os resíduos que não podem ser reciclados, compostados ou tratados de outra forma sejam descartados de maneira segura e responsável.

> A Política Nacional dos Resíduos Sólidos (Lei nº 12.305, de 2 de agosto de 2010) determina que, após a submissão dos RSU aos tratamentos e destinações disponíveis, os resíduos restantes, ou rejeitos, devam ser enviados para uma disposição final ambientalmente adequada (ABREMA, 2023, p. 26).

O local escolhido para a disposição final deve seguir normas operacionais específicas para evitar danos à saúde pública, garantir a segurança e minimizar impactos ambientais adversos (BRASIL, 2022a). O aterro sanitário é a principal instalação que se enquadra nessa definição, pois inclui medidas como impermeabilização da base, coleta e aproveitamento ou queima de biogás, drenagem e tratamento de chorume, além de monitoramento ambiental e geotécnico da área. Em contraste, lixões, aterros controlados, valas, vazadouros e áreas similares não oferecem essa proteção ambiental e são considerados inadequados para a disposição final de resíduos.

No Brasil, aproximadamente 61% dos resíduos sólidos urbanos coletados em 2022 foram destinados a aterros sanitários, totalizando 43,8 milhões de toneladas de resíduos. As áreas de disposição inadequada receberam cerca de 39% do total de resíduos coletados no mesmo período, estando presentes em todas as regiões do país.

FIGURA 9 – Disposição final de RSU no Brasil em 2022



Fonte: ABREMA (2023)

A disposição final de resíduos sólidos é a principal fonte de emissões de metano no setor, devido ao método de aterramento dos resíduos sem tratamento prévio, o que leva à decomposição anaeróbia dos materiais orgânicos e à consequente geração de metano. Portanto, a gestão integrada de resíduos pode desempenhar um papel crucial na redução das emissões de gases de efeito estufa.

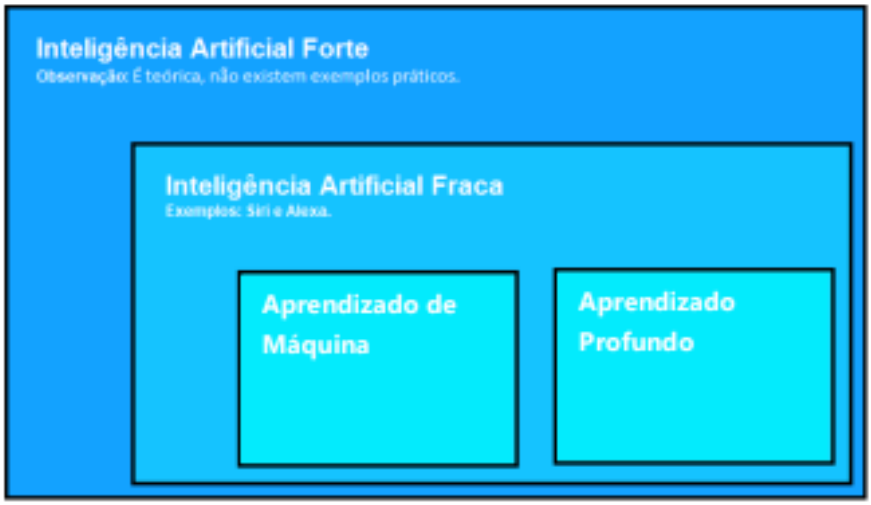
2.2 INTELIGÊNCIA ARTIFICIAL

A definição de Inteligência Artificial (IA) apresentada por Russell e Norvig (2021) descreve um campo da ciência da computação dedicado à criação de sistemas capazes de realizar tarefas que normalmente requerem inteligência humana. Essa definição destaca a natureza multifacetada do campo, que se divide em duas categorias principais:

a) **IA Fraca:** refere-se a sistemas projetados para realizar tarefas específicas e limitadas, como assistentes virtuais, algoritmos de recomendação e sistemas de reconhecimento de fala. Esses sistemas são eficazes em suas funções, mas operam dentro de um escopo restrito e não possuem a capacidade de raciocínio ou entendimento além de suas programações.
b) **IA Forte:** é uma forma teórica de IA que teria a capacidade de entender, aprender e aplicar conhecimento de maneira semelhante a um ser humano. Isso implica não apenas em realizar tarefas, mas em ter uma compreensão profunda e consciente do mundo. Embora a IA Forte ainda não tenha sido alcançada, a busca por seu desenvolvimento é um vetor importante para pesquisas contínuas nessa área, abordando questões complexas sobre a natureza da consciência, aprendizado e replicação de processos cognitivos humanos em máquinas.

Essas duas categorias refletem a diversidade de abordagens e objetivos dentro do campo da IA, desde aplicações práticas e imediatas até investigações teóricas sobre a natureza da inteligência e consciência. A IA Fraca é amplamente utilizada e já está presente em muitas aplicações do dia a dia, enquanto a IA Forte continua a ser um objetivo ambicioso e um tema de debate na comunidade científica e filosófica.

FIGURA 10 – Estruturação de uma IA



Fonte: P4Pro[1] (2021)

A Inteligência Artificial busca imitar características cognitivas humanas, como raciocínio, aprendizado, percepção, compreensão de linguagem natural e tomada de decisões (RUSSELL; NORVIG, 2021). Para alcançar esses objetivos, diversas técnicas são empregadas para aprimorar a capacidade das máquinas de resolver problemas complexos e se adaptar a novas situações. Entre as principais técnicas utilizadas estão:

a) **Aprendizado de Máquina:** desenvolve sistemas capazes de aprender, fazer previsões e tomar decisões através de dados, ajustando-se e melhorando seu desempenho ao longo do tempo. Inclui técnicas como aprendizado supervisionado, não supervisionado e por reforço.

b) **Processamento de Linguagem Natural**: dedica-se à interação entre computadores e humanos através da linguagem natural, envolvendo análise de sentimentos, tokenização, lematização e modelos de linguagem para entender e gerar texto humano de maneira coerente e contextualizada.

c) **Visão Computacional:** capacita máquinas para interpretar o mundo visual, utilizando técnicas como detecção de bordas, segmentação de imagem e redes neurais convolucionais. Essas técnicas analisam

[1] Disponível em: <https://www.p4pro.com.br/inteligencia-artificial-processos-de-negocio/>. Acesso em: 01 ago. 2024.

imagens e vídeos, possibilitando o reconhecimento de objetos e a compreensão detalhada do conteúdo visual.

d) **Robótica**: refere-se à criação de sistemas robóticos que utilizam IA para percepção, tomada de decisões e execução de tarefas físicas, permitindo a realização de atividades de forma autônoma ou quase. Esses robôs são empregados em diversas áreas, incluindo indústria, saúde, automobilismo e até em ambientes perigosos.
e) **Redes Neurais**: são modelos computacionais inspirados no cérebro humano, compostos por camadas de neurônios artificiais interconectados. As redes neurais profundas utilizam múltiplas camadas para modelar dados complexos e realizar tarefas como classificação, previsão e reconhecimento de padrões.
f) **Aprendizado Profundo:** é uma subárea do aprendizado de máquina que explora redes neurais profundas para entender e modelar padrões complexos em grandes volumes de dados. A principal vantagem dessa técnica é a sua capacidade de automatizar a extração de características, permitindo que as redes aprendam diretamente dos dados brutos e aprimorem suas representações internas ao longo do treinamento.

Cada técnica de Inteligência Artificial oferece uma abordagem única e especializada, mas frequentemente elas se combinam para criar soluções mais robustas e eficazes. A integração de métodos aprimora a capacidade da IA em enfrentar desafios complexos e promove inovações tecnológicas em diversos setores, como automação de processos, assistentes virtuais, sistemas de recomendação e interpretação de imagens.

2.3 VISÃO COMPUTACIONAL

A visão computacional, também conhecida como visão artificial, é um ramo da Inteligência Artificial que visa desenvolver sistemas com capacidades visuais semelhantes às dos seres humanos. Esses sistemas permitem que os computadores "enxerguem" e interpretem o ambiente ao seu redor, processando dados visuais, como imagens e vídeos, para extrair informações detalhadas e realizar ações utilizando algoritmos de processamento de imagem (FILHO, 2012).

FIGURA 11 – Visão Humana x Visão Computacional

Fonte: Engenharia Híbrida[2] (2022)

Este campo é multidisciplinar, integrando conhecimentos de psicologia, neurofisiologia, matemática, física e engenharia. Essa combinação possibilita a criação de sistemas para a compreensão de imagens, reconhecimento de padrões, análise e interpretação de cenas, bem como processamento óptico e de vídeo (FILHO, 2012). Algumas aplicações se estendem à ciência forense, navegação de robôs, gestão de informações, vigilância e usos militares.

O desenvolvimento de um sistema de visão computacional envolve várias etapas essenciais (NETO, 2020), dentre elas:

a) **Aquisição de Imagem:** captura de imagens ou vídeos por meio de sensores, com a configuração adequada do dispositivo para ajustar parâmetros como luminosidade, resolução e formato da imagem digital. Essa etapa é crucial para garantir a qualidade necessária para o desempenho do sistema.

b) **Pré-processamento de Imagem:** melhoria da qualidade da imagem

---

[2] Disponível em: <https://www.engenhariahibrida.com.br/post/visao-computacional-como-funciona>. Acesso em: 01 ago. 2024.

para etapas subsequentes, utilizando técnicas como atenuação de ruídos, ajuste de contraste e brilho, correção de cores e equalização de histograma.

c) **Segmentação de Imagem:** divisão da imagem em partes ou regiões de interesse. Esta etapa inclui separar objetos do fundo ou identificar áreas específicas dentro da imagem, utilizando técnicas como limiarização, detecção de bordas e agrupamento de pixels.
d) **Extração de características:** obtenção de características relevantes das regiões segmentadas para descrever e diferenciar objetos ou padrões. As características podem incluir aspectos geométricos, texturais, de cor, formato e movimento.
e) **Reconhecimento e classificação:** utilização das características extraídas para reconhecer e classificar objetos ou padrões na imagem, com o auxílio de algoritmos de aprendizado de máquina como redes neurais e máquinas de vetores de suporte.
f) **Interpretação e decisão:** interpretação dos resultados da classificação, onde o sistema toma decisões ou realiza ações baseadas nas informações processadas. Essa fase pode variar desde o simples reconhecimento de objetos até a execução de tarefas complexas, como a navegação autônoma de veículos.

A implementação específica de um sistema de visão computacional depende das funcionalidades desejadas e dos elementos de aprendizado envolvidos durante o processo. Definir claramente a funcionalidade do sistema é essencial para estabelecer a estrutura do projeto e selecionar os meios adequados.

2.4 APRENDIZADO DE MÁQUINA

O Aprendizado de Máquina, ou Machine Learning em inglês, é uma subárea da inteligência artificial que se concentra na criação de algoritmos e modelos que permitem aos computadores aprender diretamente a partir de dados. Essa capacidade possibilita que as máquinas façam previsões ou tomem decisões baseadas em padrões identificados nos dados, sem a necessidade de programação explícita para cada tarefa (RUSSELL; NORVIG, 2021).

O objetivo principal do aprendizado de máquina é a generalização, ou seja, a habilidade do modelo de aplicar o que aprendeu a novos dados. Isso é crucial para garantir que o modelo não apenas memorize os dados de treinamento, mas também possa lidar com situações não vistas anteriormente. Para avaliar essa capacidade, os dados são divididos em conjuntos de treinamento e teste. O modelo é treinado com o conjunto de treinamento e, posteriormente, avaliado com o conjunto de teste para verificar seu potencial de generalização.

Essa característica torna o aprendizado de máquina extremamente poderoso em diversas aplicações, como diagnósticos médicos, recomendações de produtos e reconhecimento de voz. Para aproveitar esse potencial, é essencial compreender os diferentes tipos de aprendizado, cada um com suas características e aplicações específicas:

a) **Supervisionado:** o modelo é treinado com um conjunto de dados rotulados, onde entradas e saídas são conhecidas, com o objetivo de aprender uma função que mapeie as entradas para as saídas corretas. É utilizado em tarefas como a classificação de e-mails e a previsão de preços imobiliários.
b) **Não Supervisionado:** o modelo é treinado com dados não rotulados, buscando identificar padrões ou estruturas subjacentes. Aplicações incluem a segmentação de clientes e a redução de dimensionalidade.
c) **Semi-supervisionado:** combina elementos dos dois tipos anteriores, utilizando um pequeno conjunto de dados rotulados e um grande conjunto de dados não rotulados. É útil quando a rotulagem de dados é cara ou demorada.
d) **Reforço:** o modelo aprende a tomar decisões através de interações com um ambiente, recebendo recompensas ou penalidades com base em suas ações. O objetivo é maximizar a recompensa total a longo prazo, sendo comumente utilizado em jogos e robótica.
e) **Transferência:** utiliza o conhecimento de uma tarefa ou domínio em outro, sendo útil quando há poucos dados para a nova tarefa, mas muitos para uma tarefa relacionada. Por exemplo, um modelo treinado em reconhecimento de imagem pode ser adaptado para classificar novas imagens.

f) **Aprendizado Profundo** (do inglês *Deep Learning*): é uma subárea que utiliza redes neurais profundas para modelar dados complexos, mostrando eficácia em tarefas como reconhecimento de imagem, processamento de linguagem natural e jogos. Exemplos incluem redes neurais convolucionais para imagens e redes neurais recorrentes para sequências de dados.

FIGURA 12 – Aprendizado de máquina

Fonte: Fabio Vivas[3] (2024)

A escolha adequada dos métodos de aprendizado e a implementação de estratégias para mitigar o *overfitting* são essenciais para o sucesso de um modelo de aprendizado de máquina. O *overfitting* ocorre quando um modelo se ajusta excessivamente aos dados de treinamento, tornando-se altamente específico e perdendo a capacidade de generalizar para novos dados. Isso compromete sua eficácia em situações do mundo real, onde a capacidade de adaptação é crucial.

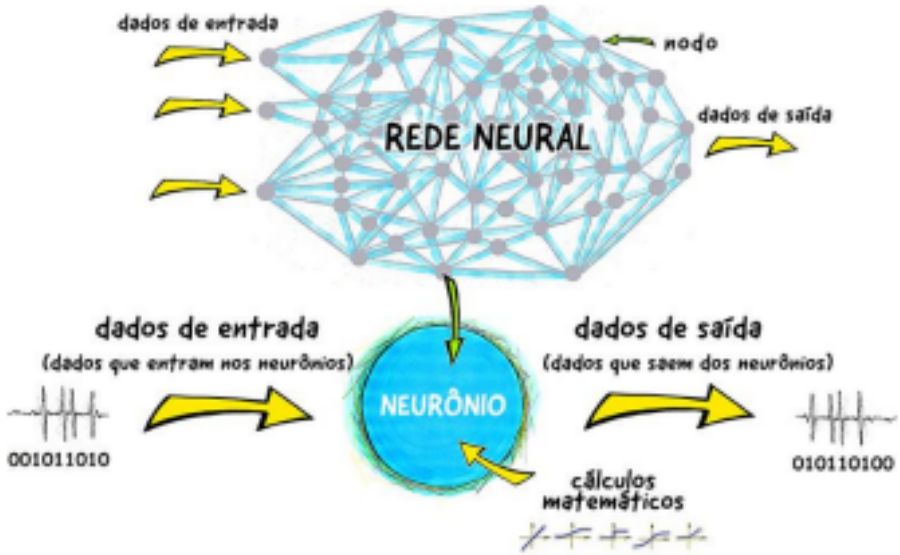
[3] Disponível em: <https://fvivas.com/aprendizado-de-maquina-algoritmos-ia/>. Acesso em: 27 ago. 2024.

Além disso, é fundamental considerar questões éticas, como o viés nos dados e a transparência dos modelos. O viés acontece quando o conjunto de treinamento reflete preconceitos, levando o modelo a tomar decisões que podem ser injustas ou discriminatórias. A transparência se refere à dificuldade de compreender como o modelo chega às suas decisões, especialmente em sistemas complexos como redes neurais profundas, o que torna mais difícil garantir que as decisões sejam justas e responsáveis.

2.5 REDE NEURAL

Segundo Simon Haykin (2009), as redes neurais são modelos computacionais inspirados na estrutura e no funcionamento do cérebro humano, projetados para reconhecer padrões, aprender a partir de dados e realizar tarefas complexas como classificação, regressão e previsão.

FIGURA 13 – Representação de uma rede neural artificial



Fonte: Unesp para Jovens[4] (2022)

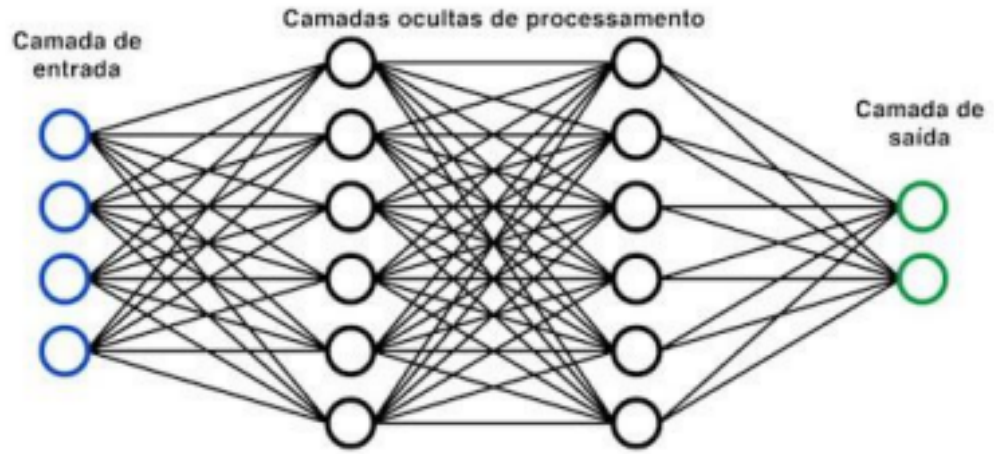
A unidade fundamental de uma rede neural é o neurônio artificial, um modelo simplificado de um neurônio biológico, cuja função é processar as entradas, aplicar uma função de ativação e gerar uma saída.

---

[4] Disponível em: <https://parajovens.unesp.br/o-que-e-uma-rede-social-e-para-que-serve/>. Acesso em: 13 set. 2024.

Os neurônios são organizados em camadas que processam as informações de forma hierárquica, ou seja, cada camada extrai características mais complexos da camada anterior e transmite esses dados para a camada seguinte. Os neurônios de diferentes camadas se comunicam entre si por meio de conexões sinápticas, representadas por pesos — valores numéricos que determinam a força e a importância do sinal transmitido.

FIGURA 14 – Camadas de uma rede neural genérica



Fonte: OPENCADD[5] (2022)

A primeira camada é a **Camada de Entrada**, responsável por receber os dados iniciais, onde cada neurônio representa uma característica ou atributo do conjunto de dados. Por exemplo, em uma rede neural projetada para classificar imagens, cada neurônio pode corresponder a um pixel da imagem.

Entre a camada de entrada e a camada de saída, há uma ou mais **Camadas Ocultas**, que realizam a maior parte do processamento ao transformar os dados de entrada em uma forma que facilita a tarefa de aprendizado. A profundidade da rede é definida pelo número dessas camadas ocultas, influenciando a capacidade do modelo de aprender representações complexas dos dados. Redes neurais profundas, com várias camadas ocultas, costumam apresentar um desempenho superior em tarefas complexas e são mais eficazes na generalização dos dados em comparação com redes mais rasas (RUSSELL; NORVIG, 2021).

---

[5] Disponivel em: <https://www.opencadd.com.br/blog/redes-neurais-e-avancos-tecnologicos>. Acesso em: 15 set. 2024.

A última camada da rede é a **Camada de Saída**, onde a saída final é gerada, podendo assumir a forma de uma classificação, regressão, previsão numérica ou outra saída, dependendo da tarefa. O número de neurônios nesta camada também varia conforme o tipo de tarefa; em uma tarefa de classificação binária, pode haver um único neurônio que indica a classe prevista, enquanto, em uma tarefa de classificação multiclasse, pode haver um neurônio para cada possível classe.

2.5.1 ARQUITETURA

As redes neurais podem ser classificadas em vários tipos, cada uma projetada para atender a diferentes tipos de problemas e aplicações. Estes são alguns dos tipos mais comuns de redes neurais:

a) ***Single Layer* Perceptron (SLP):** é o modelo mais simples de rede neural, composto por uma única camada de neurônios. É adequado para resolver problemas linearmente separáveis, ou seja, onde os dados podem ser divididos por uma linha. É utilizado em tarefas básicas de classificação, realizando uma classificação binária, na qual a saída é ativada se a soma ponderada das entradas ultrapassar um certo limiar;

b) ***Multilayer* Perceptron (MLP):** é uma extensão do perceptron, com uma ou mais camadas ocultas entre a camada de entrada e a de saída. Essa estrutura possibilita ao MLP aprender funções não lineares, tornando-o mais eficaz para lidar com problemas complexos de classificação e regressão. Os neurônios aplicam funções de ativação, permitindo que a rede capture interações complexas entre as entradas;

c) **Redes Convolucionais (CNN):** são projetadas especificamente para processamento de imagens e dados com estrutura espacial, utilizam operações de convolução para extrair características, como bordas e texturas, de forma hierárquica. São amplamente utilizadas em reconhecimento de imagem, onde a preservação da relação espacial entre pixels é essencial (HAYKIN, 2009);

d) **Redes Recorrentes (RNN):** são desenvolvidas para lidar com dados sequenciais, como texto ou séries temporais. Possuem conexões que permitem o fluxo de informações entre diferentes passos de tempo,

retendo informações de estados anteriores. Isso as torna ideais para tarefas em que o contexto prévio é fundamental, como tradução automática e análise de sentimentos;

e) **Redes Neurais de Três Dimensões (3D CNNs):** uma extensão das CNNs, projetadas para processar dados volumétricos ou sequências de imagens, como vídeos, capturando informações tanto espaciais quanto temporais (SZELISKI, 2022);
f) **Transformers**: uma arquitetura que ganhou popularidade em tarefas de linguagem natural, sendo também aplicada em visão computacional. Utilizam mecanismos de atenção para processar dados de maneira eficiente e flexível (SZELISKI, 2022);
g) **Modelos Generativos:** consistem em duas redes que competem entre si; uma gera dados enquanto a outra tenta distinguir entre dados reais e gerados, resultando em produções realistas. As *Generative Adversarial Networks* (GANs), por exemplo, são usadas na criação de imagens, vídeos e até música (SZELISKI, 2022).

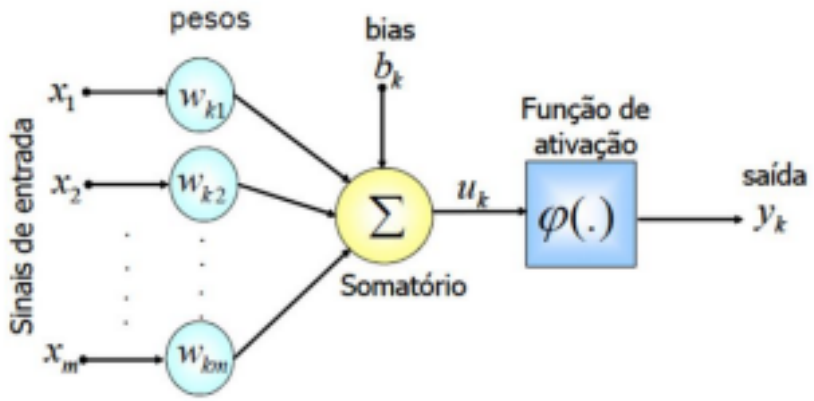
Essas redes são particularmente poderosas em tarefas que envolvem dados de alta dimensionalidade, como imagens, vídeos e sinais de fala, desempenhando um papel fundamental em aplicações como reconhecimento de voz, visão computacional e processamento de linguagem natural (HAYKIN, 2009; RUSSELL; NORVIG, 2021).

2.5.2 FUNCIONAMENTO

O funcionamento básico das redes neurais envolve a propagação dos dados da camada de entrada para a camada de saída por meio das camadas ocultas. Os neurônios recebem entradas de outros neurônios ou diretamente da camada de entrada, e cada uma dessas entradas é multiplicada por um peso, o qual determina sua importância para a saída do neurônio. Conexões com pesos altos indicam maior relevância para a decisão do neurônio, enquanto pesos baixos sugerem menor influência. Em complemento, os neurônios podem aplicar um parâmetro adicional, conhecido como bias, que ajusta a saída independentemente das entradas recebidas. Em seguida, o neurônio calcula a soma ponderada de suas

entradas, adiciona o bias e a passa por uma função de ativação. O resultado, ou saída, constitui a previsão da rede (HAYKIN, 2009).

FIGURA 15 – Esquema funcional de um neurônio artificial



Fonte: Medium[6] (2022)

As funções de ativação são essenciais, pois permitem que pequenas variações nos pesos e no bias resultem em mudanças controladas na saída do neurônio, decidindo se este deve ser ativado. Elas transformam o valor de entrada em uma saída específica, muitas vezes binária, indicando se o neurônio produzirá uma saída (SZELISKI, 2022). Além disso, introduzem não linearidade no modelo, possibilitando que a rede aprenda relações complexas nos dados e supere as limitações dos modelos lineares (HAYKIN, 2009).

A escolha da função de ativação impacta diretamente na velocidade e eficácia do treinamento da rede neural, bem como na capacidade de aprendizado e generalização a partir dos dados. As principais funções de ativação são:

a) **Função Degrau (*Heaviside*):** produz uma saída de 0 ou 1, dependendo se a entrada é menor ou maior que um limiar pré-estabelecido. Embora simples, é raramente usada em redes neurais modernas devido à sua natureza não diferenciável.
b) **Sigmoide:** transforma a entrada em um valor entre 0 e 1, seguindo a fórmula: $s\,(v) = \frac{1}{1+e^{-v}}$. É útil em problemas de classificação binária, mas pode sofrer com o "desvanecimento do gradiente" em redes profundas.

[6] Disponível em: <https://medium.com/@leticia.slopes/redes-neurais-processamento-de-linguagem-natural-29a906820e0b/>. Acesso em: 07 set. 2024.

c) **Tangente hiperbólica:** semelhante à sigmoide, mapeia a entrada para um intervalo entre -1 e 1, utilizando a fórmula: $\tanh(v) = \frac{e^{v}-e^{-v}}{e^{v}+e^{-v}}$. Sua média de saída próxima de zero pode auxiliar na convergência durante o treinamento, tornando-a uma opção frequentemente mais escolhida.

d) **ReLU (*Rectified Linear Unit*):** definida como $f(v) = \max(0, v)$, é uma das funções de ativação mais populares em redes profundas, permitindo um aprendizado eficiente e reduzindo o problema do "desvanecimento do gradiente". No entanto, pode apresentar o problema de "neurônios mortos", onde alguns neurônios deixam de aprender.

e) **Leaky ReLU:** uma variante da ReLU que ajuda a mitigar o problema dos neurônios mortos. Essa função permite uma pequena inclinação para valores negativos, definida por $f(v) = \begin{cases} v & se\ v < 0 \\ \alpha v & se\ v \geq 0 \end{cases}$, onde α é um pequeno valor positivo.

f) **Softmax:** geralmente utilizada na camada de saída em problemas de classificação multiclasse, a função softmax utiliza a fórmula $\text{softmax}(v_i) = \frac{e^{v_i}}{\sum_j e^{v_j}}$ para transformar um vetor de valores em uma distribuição de probabilidade. É ideal para situações em que a saída deve representar a probabilidade de cada classe.

2.5.3 TREINAMENTO

O treinamento de redes neurais é um processo fundamental para que os modelos aprendam com os dados e realizem previsões ou classificações precisas. O objetivo é ajustar os pesos e biases da rede para minimizar a diferença entre a saída prevista e a saída real.

Antes de iniciar o treinamento, é necessário preparar os dados, o que inclui coletar um conjunto representativo, normalizar ou padronizar os dados e dividi-los em três partes: conjunto de treinamento, conjunto de validação e conjunto de teste. O conjunto de treinamento ajusta os pesos da rede, o conjunto de validação monitora o desempenho durante o treinamento, e o conjunto de teste avalia a performance final da rede.

Inicialmente, os pesos e biases são definidos com valores geralmente aleatórios. A escolha dessa inicialização pode influenciar a velocidade e a eficácia do treinamento, além de ajudar a evitar problemas como a saturação das funções de ativação. Em seguida, para cada exemplo de entrada, a rede realiza uma propagação dos dados e gera uma saída prevista. Essa saída é comparada com a saída desejada utilizando uma função de perda, que quantifica o erro da previsão e fornece uma medida de quão bem a rede está se saindo. Comumente, funções de perda, como o erro quadrático médio, são usadas para problemas de regressão, enquanto a entropia cruzada é utilizada para problemas de classificação.

Com o erro calculado, uma retropropagação é realizada para propagar o erro de volta através da rede. Nesse processo, o gradiente da função de perda em relação aos pesos é calculado, utilizando a regra da cadeia, para determinar o ajuste necessário nos parâmetros para que o erro seja reduzido. A atualização dos pesos e biases é realizada por um algoritmo de otimização, como o gradiente descendente, e o tamanho do passo é controlado pela taxa de aprendizado. As taxas de aprendizado são hiperparâmetros que determinam a velocidade com que a rede se ajusta e podem provocar oscilações se forem muito altas, ou resultar em um treinamento lento se forem muito baixas.

Esse ciclo de propagação, cálculo da perda, retropropagação e atualização dos pesos é repetido por várias iterações até que a rede alcance um desempenho satisfatório. O conjunto de validação é utilizado para monitorar o desempenho e garantir que a rede não se ajuste excessivamente aos dados de treinamento.

Após o treinamento, a rede é avaliada com o conjunto de teste para verificar sua capacidade de generalização. Se o desempenho não for satisfatório, ajustes podem ser necessários, como modificar a arquitetura da rede, alterar a taxa de aprendizado ou aplicar técnicas de regularização. A regularização penaliza pesos muito grandes, assegurando que a rede mantenha pesos menores e mais equilibrados, o que ajuda a evitar o *overfitting*.

Uma vez treinada e avaliada, a rede pode ser implementada em aplicações do mundo real, realizando previsões com novos dados. Uma prática comum é usar modelos pré-treinados, aproveitar o conhecimento adquirido a partir de grandes conjuntos de dados e aplicá-lo a tarefas específicas (RUSSELL; NORVIG, 2021).

2.6 APRENDIZADO PROFUNDO

O Aprendizado Profundo é uma subárea do aprendizado de máquina que foca em algoritmos baseados em redes neurais artificiais com múltiplas camadas. Essas redes são capazes de aprender representações hierárquicas de dados, permitindo que o sistema extraia características de alto nível a partir de grandes volumes de dados brutos. Para isso, utilizam-se várias camadas ocultas de neurônios, onde cada camada extrai características mais complexas a partir das saídas da camada anterior, possibilitando que a rede aprenda representações cada vez mais abstratas (BISHOP, 2006; HAYKIN, 2009).

O treinamento dessas redes geralmente requer grandes quantidades de dados e significativo poder computacional, contudo, os resultados podem ser extremamente eficazes em comparação com métodos tradicionais de aprendizado de máquina (HAYKIN, 2009). Uma das vantagens do aprendizado profundo é a capacidade de aprender automaticamente características relevantes dos dados. No entanto, isso pode aumentar a propensão ao *overfitting* e comprometer a capacidade de generalização do modelo (BISHOP, 2006; SZELISKI, 2022).

O aprendizado profundo tem revolucionado o campo da inteligência artificial, resultando em avanços notáveis em diversas áreas. Sua capacidade de aprender representações complexas e de alto nível a partir de grandes volumes de dados brutos, sem a necessidade de engenharia de características manual, é um dos principais motivos para sua crescente popularidade e eficácia (BISHOP, 2006; HAYKIN, 2009; RUSSELL; NORVIG, 2021).

Entre as aplicações mais comuns do aprendizado profundo, destacam-se:

a) Reconhecimento de imagem e visão computacional, como classificação de imagens, detecção de objetos e rostos;
b) Processamento de linguagem natural, incluindo tradução automática, análise de sentimentos e geração de texto;
c) Jogos e simulações, exemplificados por sistemas como o AlphaGo;
d) Sistemas de recomendação e personalização;
e) Reconhecimento de fala, englobando conversão de fala em texto e síntese de fala.

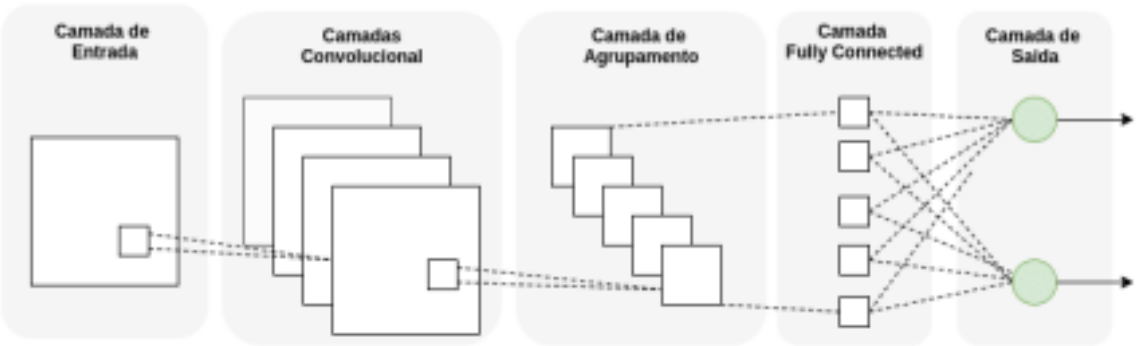
2.7 REDES CONVOLUCIONAIS

As Redes Neurais Convolucionais (CNNs) constituem uma arquitetura de rede neural desenvolvida especificamente para processar dados com estrutura de grade, como imagens, o que as torna extremamente eficazes no reconhecimento de padrões visuais. Devido à sua organização hierárquica, essas redes são utilizadas em diversas tarefas de visão computacional, incluindo detecção de objetos, reconhecimento facial e segmentação de imagens (BISHOP, 2006; HAYKIN, 2009).

A principal característica das CNNs é a presença de camadas convolucionais, que permitem à rede aprender padrões tanto espaciais quanto hierárquicos. Essas camadas aplicam filtros, ou kernels, cuja função é extrair características específicas das entradas (HAYKIN, 2009). Para melhorar a extração de informações, diversos filtros podem ser aplicados em uma única camada, permitindo capturar diferentes aspectos da imagem (RUSSELL; NORVIG, 2021).

Cada filtro é uma pequena matriz que percorre a imagem, realizando operações de convolução para detectar padrões locais. O tamanho e a quantidade de filtros são cruciais para garantir que todas as características sejam identificadas. Filtros menores, como os de 3x3 ou 5x5, são comumente utilizados, pois permitem capturar detalhes mais finos nas imagens (SZELISKI, 2022).

FIGURA 16 – Estrutura de uma Rede Neural Convolucional



Fonte: ResearchGate[7] (2021)

[7] Disponível em: <https://www.researchgate.net/figure/Figura-46-Estrutura-simplificada-de-uma-rede-neural-convolucional-Convolutional-Neural_fig6_355261609>. Acesso em: 27 nov. 2024.

As camadas iniciais são responsáveis por identificar características mais simples, como bordas e texturas, enquanto as camadas mais profundas reconhecem elementos mais complexos, como formas e objetos (SZELISKI, 2022).

O resultado dessas camadas é um conjunto de mapas de ativação, que representam a presença dessas características em diferentes regiões da imagem. Embora os mapas de ativação tenham a mesma largura e altura que a entrada, sua profundidade pode variar dependendo do número de filtros utilizados na camada convolucional (BISHOP, 2006).

Camadas de ativação geralmente são aplicadas imediatamente após as camadas convolucionais, mas podem ser consideradas como parte do processo de cálculo que ocorre em cada neurônio. Elas aplicam uma função de ativação, como ReLu, aos valores dos mapas de ativação, introduzindo não-linearidade e permitindo que a rede aprenda representações mais complexas posteriormente (HAYKIN, 2009; SZELISKI, 2022).

Posteriormente, as CNNs costumam incluir camadas de subamostragem, também conhecidas como *pooling*, que têm como objetivo reduzir a dimensionalidade dos mapas de ativação, preservando as características mais relevantes. Essa técnica é fundamental, pois diminui o número de parâmetros, reduz o tempo de computação e ajuda a controlar o *overfitting*, tornando a rede mais robusta a pequenas variações nas entradas. O tipo de *pooling* mais comum é o *max pooling*, que seleciona o valor máximo em uma região específica da imagem (RUSSELL; NORVIG, 2021).

Após várias camadas convolucionais e de *pooling*, é realizada a operação de achatamento, ou *flatten*, uma etapa crucial antes de seguirem para as camadas densas. O objetivo do *flatten* é preparar os dados para serem processados por essas camadas, transformando a saída das camadas convolucionais e de *pooling* — que geralmente são matrizes ou tensores multidimensionais — em um vetor unidimensional. Essa transformação é necessária porque as camadas densas esperam entradas no formato de vetor, onde cada elemento representa uma característica ou um valor extraído das camadas anteriores (RUSSELL; NORVIG, 2021).

Em seguida, os dados são processados por uma ou mais camadas densas, que também são chamadas de camadas totalmente conectadas (*fully connected*

layers). Nessas camadas, cada neurônio está interconectado a todos os neurônios da camada anterior, possibilitando que a rede realize a classificação final com base nas características extraídas (SZELISKI, 2022).

As Redes Neurais Convolucionais terminam com a camada de saída, onde uma função de ativação é novamente aplicada. A função *softmax* é frequentemente utilizada em problemas de classificação, pois fornece uma probabilidade para cada classe e permite que a rede faça previsões precisas. Para mitigar o *overfitting*, são aplicadas técnicas como normalização em lote (*batch normalization*) e *dropout*, que desliga aleatoriamente neurônios durante o treinamento. Essas abordagens visam garantir um desempenho superior da rede, aumentando sua robustez, capacidade de generalização e estabilidade durante o treinamento (BISHOP, 2006).

2.8 PYTHON

Criada por Guido van Rossum em 1991, o Python é uma linguagem de programação interpretada, orientada a objetos e de alto nível. Desde então, tem se destacado em áreas como desenvolvimento web, automação e análise de dados, além de mostrar grande potencial em campos mais recentes, como aprendizado de máquina, visão computacional e inteligência artificial (MARK, 2009).

Sua sintaxe clara, junto à diversidade de bibliotecas e frameworks, torna mais fácil implementar algoritmos complexos e trabalhar com grandes volumes de dados (MARK, 2009), permitindo que os profissionais foquem na solução dos problemas, ao invés de se preocuparem com os detalhes técnicos. Essa simplicidade e legibilidade fizeram dela uma escolha popular entre desenvolvedores, pesquisadores e cientistas de dados (SZELISKI, 2022).

FIGURA 17 – Logotipo Python

Fonte: Python[8] (2024)

[8] Disponível em: <https://www.python.org/>. Acesso em: 09 out. 2024.

Com o surgimento de bibliotecas como OpenCV, TensorFlow, Pandas, Keras e PyTorch, além da crescente demanda por soluções de inteligência artificial, o Python se consolidou como a linguagem preferida para o desenvolvimento de aplicações que envolvem análise visual e interpretação de dados (GERON, 2019; MARK, 2009). Essas bibliotecas fornecem ferramentas robustas para o processamento de imagens, treinamento de modelos de aprendizado profundo e implementação de Redes Neurais Convolucionais. A facilidade de integração dessas ferramentas com Python possibilita que desenvolvedores criem aplicações avançadas, como o desenvolvimento e treinamento de modelos CNN, ao mesmo tempo em que simplifica a criação de protótipos e a experimentação (MARK, 2009; SZELISKI, 2022).

> A popularidade de Python se deve à sua comunidade ativa, vasta gama de bibliotecas e frameworks, e à sua capacidade de ser utilizado em diferentes domínios, tornando-o uma escolha ideal tanto para iniciantes quanto para desenvolvedores experientes (MARK, 2009, p. 63).

Por meio de bibliotecas como NumPy e Matplotlib, desenvolvedores conseguem manipular dados e visualizar resultados de forma eficiente, o que facilita a análise e interpretação deles. Além disso, a comunidade ativa do Python, juntamente com o suporte contínuo, garante o acesso às técnicas mais recentes por meio da constante contribuição com novos pacotes e atualizações regulares (SZELISKI, 2022). Essa dinâmica mantém o Python na vanguarda da inovação em visão computacional e aprendizado de máquina, impulsionando ainda mais suas aplicações em projetos acadêmicos e industriais (MARK, 2009).

3 TRABALHOS RELACIONADOS

Este capítulo tem como objetivo apresentar trabalhos com propostas semelhantes a este, focando em visão computacional e classificação de lixo. Serão abordadas as tecnologias, metodologias, linguagens utilizadas, além dos desafios e soluções mais relevantes.

3.1 UTILIZAÇÃO DE BIBLIOTECAS DE VISÃO COMPUTACIONAL E APRENDIZADO DE MÁQUINA NA IDENTIFICAÇÃO DE RESÍDUOS SÓLIDOS (METAL E VIDRO) APLICADO A COLETA SELETIVA

O estudo conduzido por Pina e Miranda (2018) teve como propósito o desenvolvimento e a implementação de um sistema automatizado para classificar resíduos sólidos, especificamente metal e vidro. A implementação foi realizada em Python, integrando as bibliotecas OpenCV e TensorFlow para viabilizar a classificação em tempo real desses resíduos. Considerou-se também a inclusão de uma interface gráfica para facilitar a interação do usuário com o sistema.

Na primeira etapa, a coleta e preparação dos dados foi realizada por meio da captura de imagens de resíduos sólidos sob diferentes condições de iluminação e ângulos, assegurando a diversidade do conjunto de dados. A rotulagem das imagens utilizou a ferramenta LabelImg, responsável por gerar arquivos XML com informações sobre as classes e as delimitações dos objetos identificados.

FIGURA 18 – Mapeamento de imagem com LabelImg

Fonte: PINA e MIRANDA (2018)

Após essa fase, as imagens passaram por um pré-processamento para otimizar a convergência durante o treinamento, incluindo redimensionamento para um tamanho uniforme, adequado ao modelo, e normalização dos valores de pixels na faixa de [0, 1]. Técnicas de aumento de dados, como rotação, translação,

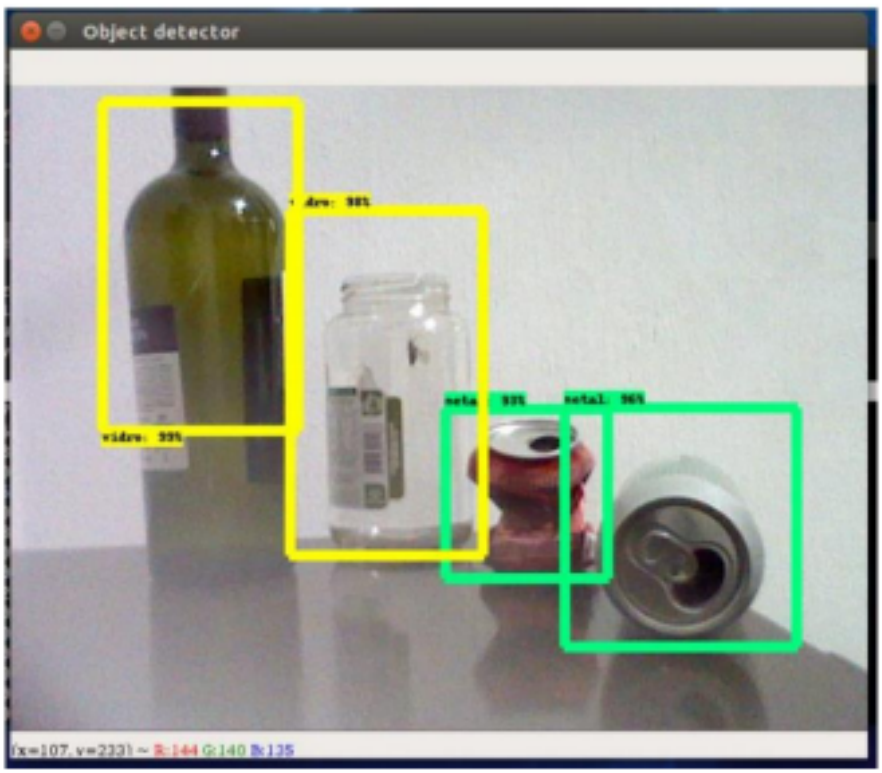
inversão horizontal e variação de brilho, foram aplicadas para ampliar a variabilidade do conjunto e reduzir o risco de *overfitting*.

Na etapa de modelagem e treinamento, foi utilizada a biblioteca TensorFlow, com ênfase em redes neurais convolucionais (CNNs), e a arquitetura *Faster* R-CNN, que é projetada para detectar objetos em imagens de maneira precisa, identificando a classe e a localização através de caixas delimitadoras.

seu desempenho eficaz em classificação de imagens e pela otimização para dispositivos de recursos limitados. O conjunto de dados foi dividido em 70% para treinamento, realizado com o otimizador Adam e a função de perda de entropia cruzada, 15% para validação e 15% para teste.

Para avaliar o modelo, utilizaram-se métricas como acurácia e precisão, além do tempo de resposta, que analisou o desempenho nas classes de resíduos metal e vidro. Com base nos resultados de validação, ajustes nos hiperparâmetros, como a taxa de aprendizado e o número de épocas, foram realizados para aprimorar o modelo.

FIGURA 19 – Teste com múltiplos objetos



Fonte: PINA e MIRANDA (2018)

Durante o desenvolvimento do sistema, surgiram desafios como a necessidade de um grande volume de dados rotulados. Para atenuar essas dificuldades, aplicaram-se técnicas como aumento de dados e transferência de aprendizado, aproveitando modelos pré-treinados adaptados à tarefa específica de classificação de resíduos.

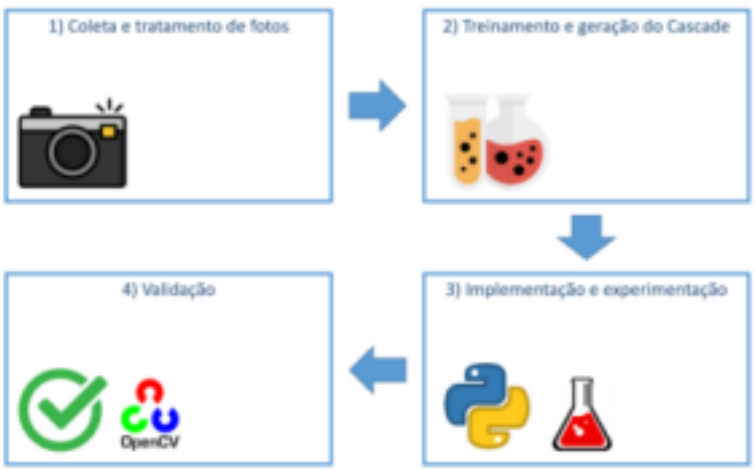
O projeto resultou em um sistema funcional para identificação e classificação de resíduos sólidos, promovendo o uso de aprendizado de máquina e visão computacional na gestão de resíduos, com aplicações inovadoras e sustentáveis.

3.2 PROVA DE CONCEITO (PoC) PARA DETECÇÃO DE OBJETOS USANDO VISÃO COMPUTACIONAL EM UMA COLETA SELETIVA

A pesquisa realizada por Oliveira e Stoll (2023) teve como objetivo desenvolver um sistema de detecção de materiais recicláveis, como plásticos e metais, utilizando técnicas de reconhecimento de imagem. A motivação central foi automatizar a coleta seletiva para melhorar a eficiência do processo e reduzir a exposição dos trabalhadores a materiais potencialmente perigosos.

A primeira etapa de desenvolvimento foi definir a metodologia, composta pela linguagem Python, que se destaca pela simplicidade e variedade de bibliotecas como TensorFlow e scikit-learn. A biblioteca OpenCV também foi selecionada, dada sua robustez e ampla aceitação na comunidade de desenvolvedores.

FIGURA 20 – Topologia do projeto



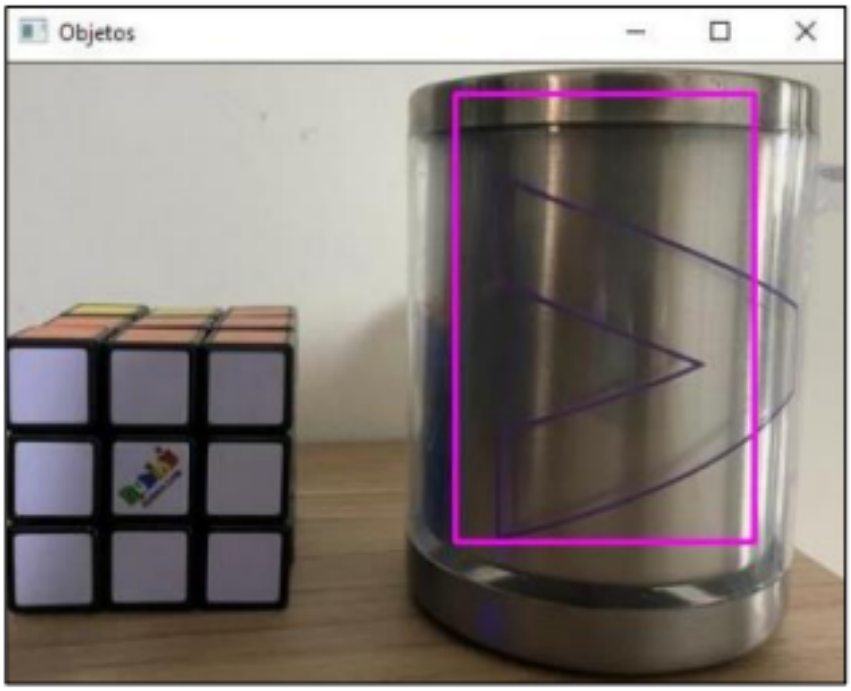
Fonte: OLIVEIRA e STOLL (2023)

Na fase de coleta de dados, foi criado um conjunto diversificado de imagens de objetos recicláveis, rotuladas manualmente para compor o *dataset* necessário ao treinamento do modelo. O pré-processamento das imagens incluiu o redimensionamento, a normalização de cores e o aumento de dados, que reforçam a robustez do modelo diante das variações de iluminação e ângulos de captura.

O treinamento foi conduzido com o uso do *Cascade Trainer GUI*, uma ferramenta que permite a criação de classificadores em cascata para a detecção de objetos. O modelo foi alimentado com um conjunto de imagens positivas (objetos recicláveis) e negativas (objetos sem interesse), utilizando o classificador em cascata de Haar, que demonstrou eficiência para detecção em tempo real.

Após o treinamento, a avaliação do modelo foi realizada com um conjunto de testes separado, aplicando métricas como acurácia e precisão. O modelo obteve uma acurácia de 80%, o que representa um desempenho satisfatório para o contexto da aplicação proposta.

FIGURA 21 – Validação dos dados



Fonte: OLIVEIRA e STOLL (2023)

A implementação final integrou o modelo treinado em uma aplicação Python capaz de capturar imagens em tempo real com o uso de uma câmera, processar

as imagens, aplicar o modelo de detecção e classificar os objetos de maneira contínua. Esse sistema foi desenvolvido em uma arquitetura modular, que favorece expansões e aprimoramentos futuros.

Entre os principais desafios encontrados, destacaram-se a variabilidade das condições de iluminação e a diversidade de objetos, que impactaram a precisão da classificação. Para minimizar esses problemas, foram aplicadas técnicas de pré-processamento, como equalização de histograma e filtragem de ruído, que melhoraram a qualidade das imagens antes da detecção. Também foi considerada a implementação de um sistema de feedback para ajustes em tempo real nos parâmetros do modelo, como possibilidade para futuras iterações do projeto.

Em conclusão, o estudo demonstra que a combinação de tecnologias de visão computacional e metodologias de aprendizado de máquina oferece soluções inovadoras para problemas ambientais atuais, especialmente em gestão de resíduos e reciclagem. A pesquisa valida a proposta inicial e apresenta um ponto de partida relevante para estudos futuros na área de sustentabilidade.

3.3 ANÁLISE COMPARATIVA

TABELA 1 – Análise comparativa dos projetos

PARÂMETROS	PROJETO 1	PROJETO 2
Arquitetura	*Faster* R-CNN Pré-treinado	Classificador HAAR-*Cascade* Pré-treinado
Objeto-alvo	Resíduos sólidos (metal e vidro)	Materiais Recicláveis (plásticos e metais)
Dataset	Captura em tempo real	Próprio: 20 imagens/objeto Captura em tempo real
Métricas	Acurácia – Metal: 66.66% Acurácia – Vidro: 100% Precisão – Metal: 91–98% Precisão – Vidro: 85–99%	Acurácia: 80% Precisão: 80%

Fonte: Autora

O **Projeto 1**, baseado na *Faster* R-CNN pré-treinada, alimentado por imagens em tempo real, combina alta acurácia e tempo de resposta eficiente, ideal para classificações complexas, mas exige alto poder computacional.

O **Projeto 2** utiliza o classificador Haar-*Cascade* pré-treinado, inicialmente com imagens capturadas, oferecendo análise controlada e desempenho equilibrado em tarefas de média complexidade, mas vulnerável a interferências ambientais com imagens em tempo real.

4 MATERIAIS E MÉTODOS

Este capítulo descreve os materiais e ferramentas utilizados no desenvolvimento do projeto, incluindo os softwares e bibliotecas empregadas. Também são apresentadas as etapas do processo, como a coleta e preparação dos dados, a configuração do ambiente de desenvolvimento e a execução dos experimentos. Além disso, as escolhas metodológicas adotadas são justificadas, bem como os ajustes realizados nos modelos ao longo do desenvolvimento.

4.1 MATERIAIS

Para o desenvolvimento deste projeto, foram escolhidas as plataformas *Kaggle* e *Google Colab*, amplamente reconhecidas por sua relevância em projetos relacionados a aprendizado de máquina e ciência de dados.

FIGURA 22 – Logotipo do *Kaggle*

Fonte: Wikipedia[9] (2024)

O *Kaggle* é uma plataforma voltada para o compartilhamento de *datasets*, participação em competições e aprendizado colaborativo. Ela oferece acesso a uma imensa variedade de bases de dados e é amplamente utilizada por pesquisadores e profissionais da área de ciência de dados.

Para este projeto, foram utilizados dois *datasets* disponibilizados no *Kaggle*,

[9] Disponível em: <https://en.wikipedia.org/wiki/File:Kaggle_logo.png>. Acesso em: 17 nov. 2024.

ambos compostos por imagens de resíduos sólidos. O primeiro[10], denominado *Garbage Classification* e disponibilizado pelo usuário cchangcs, apresenta seis classes e se destaca como o *dataset* mais popular na área, com o maior número de *upvotes* e mais de 40 mil downloads ao longo de seis anos.

O *dataset*, intitulado *Garbage Classification (12 classes)* e criado por Mostafa Mohamed, é um dos mais relevantes na área de resíduos sólidos, ocupando a segunda posição na classificação do *Kaggle*. Foi desenvolvido com a premissa de que muitos *datasets* na área classificam os resíduos em apenas 2 a 6 classes, e que o aumento do número de classes poderia contribuir significativamente para o avanço da reciclagem de resíduos domésticos.

Embora o *dataset* original de Mostafa seja completo e adequado, uma versão aprimorada foi disponibilizada por Hubert Hamelin, intitulada *Garbage Classification (12 classes) ENHANCED*. Essa versão apresenta melhorias substanciais, como a correção de rótulos, a remoção de imagens duplicadas e a adição de novas imagens, resultando em um conjunto de dados mais robusto e confiável, o que justifica sua escolha como o segundo[11] *dataset* utilizado no desenvolvimento deste projeto.

Apesar da quantidade de classes disponíveis nos *datasets* selecionados, este projeto concentrou-se em apenas quatro categorias específicas: papel, plástico, vidro e metal. A distribuição das imagens por classe foi a seguinte: vidro (1798 imagens), metal (1657 imagens), plástico (1865 imagens) e papel (1799 imagens), totalizando 7119 imagens.

FIGURA 23 – Logotipo Google Colab

Fonte: Colab[12] (2024)

[10] Disponível em: <https://www.kaggle.com/datasets/asdasdasasdas/garbage-classification/data>. Acesso em: 07 ago. 2024.
[11] Disponível em: <https://www.kaggle.com/datasets/huberthamelin/garbage-classification-labels-corrections>. Acesso em: 07 ago. 2024.
[12] Disponível em: <https://colab.google/>. Acesso em: 17 nov. 2024.

O *Google Colaboratory*, por sua vez, é uma ferramenta baseada na nuvem que permite a execução de notebooks Jupyter diretamente em um ambiente web. Ele oferece suporte completo à linguagem Python e a diversas bibliotecas populares de aprendizado de máquina, proporcionando um ambiente acessível e eficiente para o desenvolvimento das soluções propostas.

Essa plataforma proporciona acesso gratuito a recursos computacionais avançados, como GPUs e TPUs, dispensando a necessidade de configuração local. Isso facilita o processamento de grandes volumes de dados e acelera a execução de treinamentos para modelos mais complexos e otimizados. Além disso, sua interface colaborativa e a integração com o Google Drive tornam o compartilhamento e o armazenamento de projetos mais práticos e eficientes.

No desenvolvimento do projeto, diversas bibliotecas foram utilizadas para integrar funcionalidades e recursos à linguagem Python. Abaixo estão as principais e suas respectivas funcionalidades:

a) **OS** é utilizada para manipulação e navegação de diretórios e arquivos no sistema operacional;
b) **Google.colab.drive** integra o Google Colab com o Google Drive, facilitando o acesso e o armazenamento de arquivos diretamente na nuvem;
c) **Pandas** simplifica a manipulação e análise de dados estruturados, como tabelas e dataframes;
d) **Numpy** é responsável por realizar cálculos matemáticos e manipular arrays multidimensionais;
e) **Random** gera números aleatórios, útil para selecionar imagens dos datasets;
f) **Skimage** redimensiona imagens para padronizar seus tamanhos antes de alimentá-las ao modelo;
g) **Matplotlib** é usada para plotar gráficos e exibir imagens durante a análise dos datasets;
h) **Seaborn** complementa o Matplotlib, gerando gráficos estatísticos, como matrizes de confusão, para a análise dos resultados;
i) **TensorFlow** e **Keras** são as bibliotecas principais utilizadas para

construir, treinar e validar modelos de aprendizado profundo. Elas oferecem diversas ferramentas como:

- **ImageDataGenerator**, que aumenta os dados e prepara lotes de imagens;
- **EarlyStopping**, que interrompe o treinamento ao detectar estagnação na performance do modelo;
- **MobileNetV2**, uma arquitetura pré-treinada eficiente, usada para aprendizado por transferência;
- Classes como **Sequential**, **Model** e **Layers** são fundamentais para a construção de redes neurais personalizadas;
- Camadas como **Conv2D**, **MaxPooling2D**, **BatchNormalization**, **Dropout**, **Flatten**, **Dense**, **Input** e **Activation** são aplicadas para criar, implementar e otimizar redes neurais convolucionais.

j) **Scikit-learn** é uma biblioteca que avalia o desempenho dos modelos, oferecendo ferramentas como:

- a função **train_test_split** é usada para dividir os dados em conjuntos de treinamento, validação e teste, garantindo uma avaliação robusta e consistente do modelo.
- o **classification_report** e a **confusion_matrix**, que fornecem métricas detalhadas como precisão, revocação e F1-score, essenciais para entender a eficácia do modelo;

Essas bibliotecas desempenharam um papel crucial no desenvolvimento do projeto, possibilitando o processamento de dados, a construção e o treinamento dos modelos, além da avaliação dos resultados obtidos. Também permitiram que todas as etapas de concepção fossem executadas de forma eficiente e bem estruturada.

4.2 MÉTODOS

Esta seção descreve o desenvolvimento e a implementação dos modelos de redes neurais convolucionais utilizados para a classificação de resíduos em quatro categorias. Os modelos foram projetados com diferentes arquiteturas, explorando abordagens baseadas em aprendizado por transferência e redes totalmente

customizadas. Abaixo são apresentadas as etapas de execução, bem como a descrição detalhada de cada modelo desenvolvido.

4.2.1 PARÂMETROS

Para garantir consistência durante o desenvolvimento e treinamento, foram adotados os seguintes parâmetros globais:

a) **Dimensão das imagens:** 224 x 224 x 3 pixels, em conformidade com os requisitos de modelos pré-treinados;
b) **Número de classes:** 4, correspondentes às categorias de resíduos;
c) **Tamanho de lote (*batch size*):** 64, visando balancear a eficiência computacional e a estabilidade do treinamento;
d) **Número de épocas:** 50, buscando alcançar uma convergência satisfatória;
e) **Semente (*seed*):** 42, para garantir a reprodutibilidade dos resultados.

FIGURA 24 – Parâmetros globais

```
image_size = (224, 224)
input_shape = (224, 224, 3)
num_classes = 4
batch_size = 64
epochs = 50
seed = 42
```

Fonte: Autora

4.2.2 DIVISÃO DOS CONJUNTOS

Para treinar os modelos propostos de forma eficaz foi necessário dividir o conjunto de dados em três subconjuntos: treinamento com 70%, validação com 15% e teste também com 15%. Essa divisão permite que o modelo tenha uma base sólida para o aprendizado inicial, um conjunto específico para o ajuste fino de hiperparâmetros e outro destinado à avaliação final de desempenho. Isso garante uma análise confiável dos resultados obtidos e contribui para melhorar a capacidade dos modelos em se adaptarem a novos conjuntos de dados.

FIGURA 25 – Função para dividir os conjuntos de dados

```python
def DataFrameSpliting(df, train_ratio, val_ratio, test_ratio):

    trainDf, testDf = train_test_split(df, test_size=test_ratio, stratify=df['label'], random_state=seed)

    trainDf, valDf = train_test_split(trainDf, test_size=val_ratio / (train_ratio + val_ratio), stratify=trainDf['label'], random_state=seed)

    trainDf = trainDf.sample(frac=1).reset_index(drop=True)
    valDf = valDf.sample(frac=1).reset_index(drop=True)
    testDf = testDf.sample(frac=1).reset_index(drop=True)

    return trainDf, valDf, testDf
```

Fonte: Autora

A divisão dos dados foi realizada de forma estratificada, garantindo que a distribuição das classes fosse preservada em cada subconjunto. O procedimento seguiu os passos descritos abaixo:

1) **Divisão inicial:** o conjunto foi dividido em treinamento e teste, respeitando a proporção definida e utilizando a variável de classe como critério de estratificação;
2) **Subdivisão do treinamento:** o conjunto de treinamento foi posteriormente dividido em treinamento e validação, ajustando a proporção entre essas partes;
3) **Reordenação e redefinição de índices:** os dados foram reordenados e seus índices redefinidos, assegurando que não houvesse sobreposição entre os subconjuntos;
4) **Implementação automatizada:** a função DataFrameSpliting foi empregada para realizar a divisão de forma sistemática, recebendo como parâmetros o dataframe original e as porcentagens desejadas para cada conjunto;
5) **Verificação da distribuição:** A distribuição das amostras por classe foi confirmada por meio de contagens detalhadas e gráficos, assegurando equilíbrio para o treinamento.

FIGURA 26 – Implementação e verificação da divisão dos dados

```python
trainDf, valDf, testDf = DataFrameSpliting(df, 0.70, 0.15, 0.15)
print("Training DataFrame = " + str(len(trainDf)) + " images")
print("Validation DataFrame = " + str(len(valDf)) + " images")
print("Testing DataFrame = " + str(len(testDf)) + " images")
```

Fonte: Autora

4.2.3 PRÉ-PROCESSAMENTO

O pré-processamento dos dados foi realizado para assegurar a uniformidade das imagens e facilitar o aprendizado dos modelos. Os passos seguidos foram:

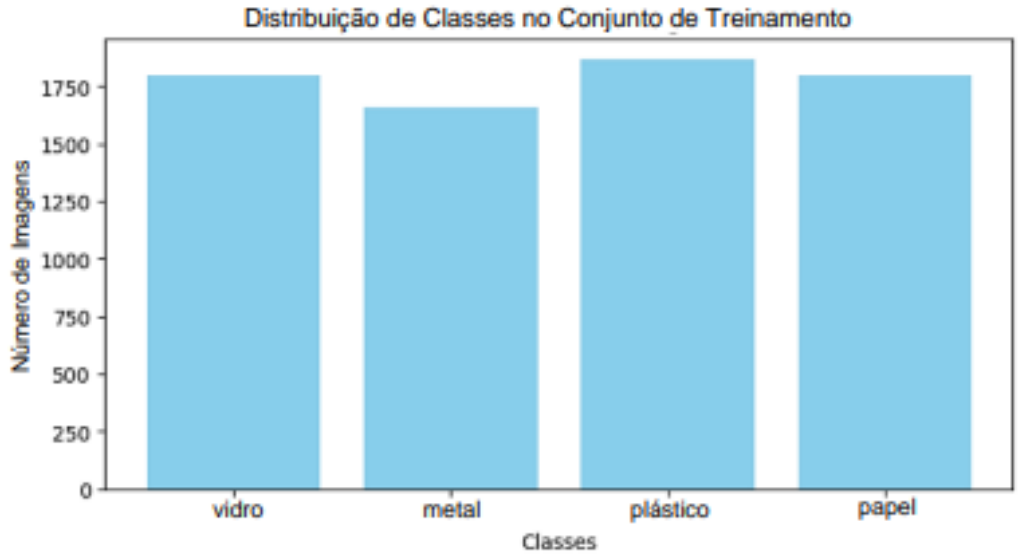
1) **Redimensionamento:** todas as imagens foram redimensionadas para 224x224 pixels, com três canais de cor (RGB), atendendo as especificações do modelo pré-treinado MobileNetV2;
2) **Inspeção visual:** amostras representativas do conjunto de dados foram exibidas em um *grid*, permitindo a verificação da qualidade das imagens e a identificação de possíveis problemas, como resolução inadequada ou desequilíbrios na distribuição das classes;

FIGURA 27 – Amostragem de imagens

Fonte: Autora

3) **Contagem e distribuição:** a quantidade de imagens por classe foi calculada e exibida em um gráfico de barras. Essa etapa possibilitou avaliar a distribuição das classes, verificando o equilíbrio entre elas, e aplicar técnicas de balanceamento, caso necessário.

FIGURA 28 – Gráfico de distribuição de classes



Fonte: Autora

4.2.4 AUMENTO DE DADOS

Para reduzir o risco de *overfitting* e aumentar a diversidade dos dados de treinamento, foi aplicada a técnica de *data augmentation*. Essa estratégia consiste em transformar as imagens existentes por meio de operações geométricas e de intensidade, gerando variações que simulam novas amostras.

FIGURA 29 – Aumento de dados no conjunto de Treinamento

```
datagenTrain = ImageDataGenerator(
            rescale = 1./255,
            zoom_range = (1.0, 1.2),
            horizontal_flip = True,
            vertical_flip = True,
            rotation_range = 45,
)

trainGenerator = datagenTrain.flow_from_dataframe(
    trainDf,
    x_col = 'imgPath',
    y_col = 'label',
    target_size = image_size,
    batch_size = batch_size,
    class_mode = 'categorical'
)
```

Fonte: Autora

As transformações utilizadas incluem:

1) **Redimensionamento:** normalização dos valores de pixel para o intervalo [0,1];
2) **Zoom:** variações de escala com fator entre 1.0 e 1.2;
3) ***Flip* horizontal e vertical:** inversão das imagens ao longo dos eixos horizontal e vertical;
4) **Rotação:** alterações de orientação com ângulos aleatórios de até 45°.

O aumento de dados foi configurado exclusivamente para o conjunto de treinamento, garantindo uma avaliação imparcial do desempenho dos modelos. As imagens aumentadas foram organizadas em lotes (*batches*) de 64 amostras para otimizar o uso de memória durante o treinamento.

Nos conjuntos de validação e teste, foi realizada apenas a normalização dos valores de pixel e organização dos dados em lotes de 8 amostras, com o embaralhamento (*shuffle*) desativado para evitar alteração na ordem das amostras e garantir a reprodutibilidade dos resultados.

FIGURA 30 – Aumento de dados nos conjuntos de Validação e Teste

```
datagenVal = ImageDataGenerator(rescale=1./255)

valGenerator = datagenVal.flow_from_dataframe(
    valDf,
    x_col = 'imgPath',
    y_col = 'label',
    target_size = image_size,
    batch_size = 8,
    class_mode = 'categorical',
    shuffle = False
)

datagenTest = ImageDataGenerator(rescale=1./255)

testGenerator = datagenTest.flow_from_dataframe(
    testDf,
    x_col = 'imgPath',
    y_col = 'label',
    target_size = image_size,
    batch_size = 8,
    class_mode = 'categorical',
    shuffle = False
)
```

Fonte: Autora

4.2.5 MODELOS PROPOSTOS

O **Modelo 0** adota o aprendizado por transferência através da arquitetura MobileNetV2 com pesos pré-treinados no conjunto ImageNet. As primeiras camadas convolucionais foram congeladas para preservar os pesos pré-treinados, enquanto as últimas foram ajustadas para a classificação. Este modelo inclui:

a) Uma camada *Flatten* para transformar a saída convolucional em um vetor unidimensional.
b) Uma camada densa com 64 neurônios e ativação ReLU;
c) *Batch Normalization* para melhorar a estabilidade do treinamento;
d) *Dropout* com taxa de 0,08 para mitigar o *overfitting*;
e) Uma camada de saída com função de ativação Softmax.

FIGURA 31 – Arquitetura do Modelo 0

```
Model0 = Sequential([
    Input(shape=input_shape),

    MobileNetV2(weights='imagenet', include_top=False, input_shape=input_shape),

    Flatten(),

    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.08),

    Dense(num_classes, activation='softmax')
])

preTrainedModel0 = Model0.layers[0]
for layer in preTrainedModel0.layers[:-4]:
    layer.trainable = False

Model0.compile(optimizer='adam',
               loss='categorical_crossentropy',
               metrics=['accuracy']
               )
Model0.summary()
```

Fonte: Autora

O **Modelo 1** apresenta uma configuração semelhante ao Modelo 0, com diferenças no mecanismo de regularização. Este modelo se distingue pelo uso de

uma taxa de *Dropout* de 0,2, o que aumenta a robustez ao *overfitting*.

FIGURA 32 – Arquitetura do Modelo 1

```
Model1 = Sequential([
    Input(shape=input_shape),

    MobileNetV2(weights='imagenet', include_top=False, input_shape=input_shape),

    Flatten(),

    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),

    Dense(num_classes, activation='softmax')
])
```

Fonte: Autora

O **Modelo 2** também utiliza a arquitetura MobileNetV2, mas incorpora uma camada densa intermediária maior. Suas principais características incluem:

a) Camada densa com 128 neurônios e ativação ReLU;
b) *Dropout* com taxa de 0,08 para regularização.

FIGURA 33 – Arquitetura do Modelo 2

```
Model2 = Sequential([
    Input(shape=input_shape),

    MobileNetV2(weights='imagenet', include_top=False, input_shape=input_shape),

    Flatten(),

    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.08),

    Dense(num_classes, activation='softmax')
])
```

Fonte: Autora

O **Modelo 3** mantém o *Dropout* e adota camadas densas múltiplas para capturar melhor as relações entre as classes. Sua principal característica inclui:

a) Camada densa com 64 neurônios, seguida de uma camada com 32 neurônios, ambas ativadas por ReLU;

FIGURA 34 – Arquitetura do Modelo 3

```
Model3 = Sequential([
    Input(shape=input_shape),

    MobileNetV2(weights='imagenet', include_top=False, input_shape=input_shape),

    Flatten(),

    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    BatchNormalization(),
    Dropout(0.08),

    Dense(num_classes, activation='softmax')
])
```

Fonte: Autora

O **Modelo 4** foi desenvolvido do zero, com quatro blocos convolucionais para extração de características. Cada bloco inclui:

a) Uma camada convolucional, com diferentes números de neurônios, seguida de ativação ReLU;
b) Uma camada de *pooling* para redução da dimensionalidade;

FIGURA 35 – Arquitetura convolucional do Modelo 4

```
Model4 = Sequential([
    Input(shape=input_shape),

    Conv2D(32, 3),
    Activation(activation="relu"),
    MaxPooling2D(),

    Conv2D(64, 3),
    Activation(activation="relu"),
    MaxPooling2D(),

    Conv2D(128, 3),
    Activation(activation="relu"),
    MaxPooling2D(),

    Conv2D(128, 3),
    Activation(activation="relu"),
    MaxPooling2D(),
```

Fonte: Autora

Após os blocos convolucionais, o modelo segue para a parte densa. As principais características desta parte incluem:

a) Uma camada *Flatten*
b) Duas camadas densas de 128 neurônios cada;
c) *Dropout* de 0,2;
d) Uma camada de saída Softmax para classificação.

FIGURA 36 – Arquitetura densa do Modelo 4

```
    Flatten(),

    Dense(128, activation='relu'),
    Dense(128, activation='relu'),
    Dropout(0.2),

    Dense(num_classes, activation='softmax')
])
```

Fonte: Autora

4.2.6 TREINAMENTO E AVALIAÇÃO

Os cinco modelos foram treinados e avaliados com os mesmos parâmetros globais e conjunto de dados. O treinamento seguiu o código base do Modelo 0, utilizando geradores de dados (*trainGenerator* e *valGenerator*) para os conjuntos de treinamento e validação. O número de épocas foi determinado pela variável *epochs*, e o processo foi monitorado com *callbacks* de *EarlyStopping*, interrompendo o treinamento caso a acurácia de validação não melhorasse após um número pré-determinado de épocas.

FIGURA 37 – Treinamento dos modelos

```
history0 = Model0.fit(trainGenerator,
                      validation_data = valGenerator,
                      epochs = epochs,
                      verbose = 1,
                      callbacks = [EarlyStopping(
                                        patience = 4,
                                        monitor = 'val_accuracy',
                                        restore_best_weights = True)
                      ])
```

Fonte: Autora

Para a avaliação de cada modelo, foi implementada a função calculate_evaluation. Esta função calcula a perda (*loss*) e a acurácia (*accuracy*) de cada modelo utilizando o conjunto de teste. Além disso, gera a Matriz de Confusão e o Relatório de Classificação, com métricas como acurácia, precisão, *recall* e F1-*score*. Métricas adicionais, como Sensibilidade e Especificidade, também foram calculadas a partir dos valores da matriz de confusão, fornecendo uma análise mais detalhada do desempenho dos modelos.

FIGURA 38 – Avaliação dos modelos

```
def calculate_evaluation(model, model_name):

    loss, accuracy = model.evaluate(testGenerator, verbose = 0)

    predictions = model.predict(testGenerator, verbose = 0)
    true_labels = testGenerator.classes
    true_labels[:10]
    predicted_labels = predictions.argmax(axis=-1)
    predicted_labels[:10]

    print(f"_______ EVALUATION - {model_name} _______\n")

    print(f"------ CONFUSION MATRIX ------\n")
    cm = confusion_matrix(true_labels, predicted_labels)
    sns.heatmap(cm, center = True,cmap='terrain',annot=True ,fmt='.5g')
    plt.show()

    ClassificationReport = classification_report(true_labels, predicted_labels)
    print(f"\n------ CLASSIFICATION REPORT ------\n\n", ClassificationReport)

    FP = cm.sum(axis=0) - np.diag(cm)
    FN = cm.sum(axis=1) - np.diag(cm)
    TP = np.diag(cm)
    TN = cm.sum() - (FP + FN + TP)

    sensitivity = np.mean(TP / (TP + FN))
    specificity = np.mean(TN / (TN + FP))

    print(f"\nLoss: {loss:.4f}  |  Accuracy: {accuracy:.4f}")
    print(f"\nSensitivity: {sensitivity:.4f}  |  Specificity: {specificity:.4f}\n")


calculate_evaluation(Model0, "Model 0")
calculate_evaluation(Model1, "Model 1")
calculate_evaluation(Model2, "Model 2")
calculate_evaluation(Model3, "Model 3")
calculate_evaluation(Model4, "Model 4")
```

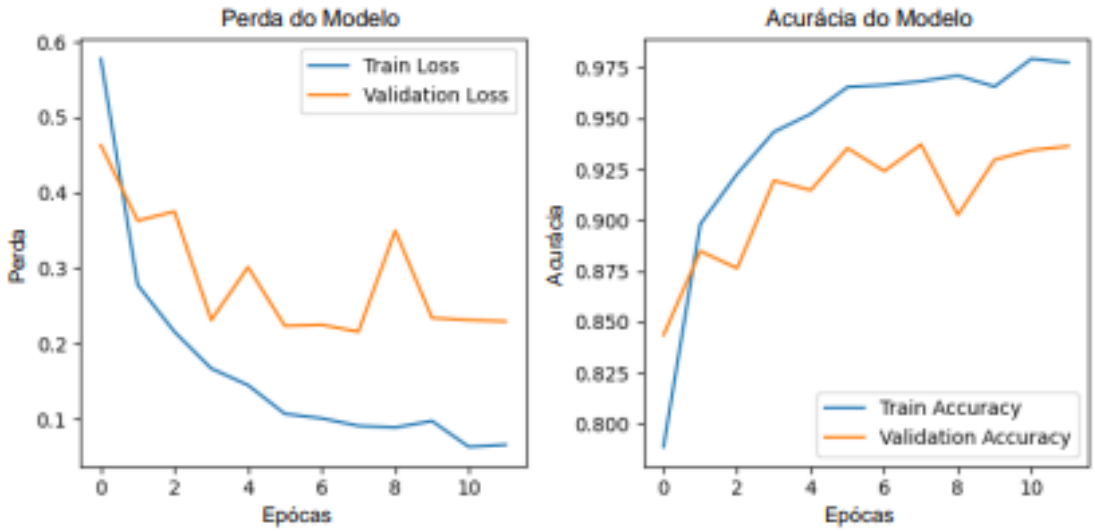
Fonte: Autora

Além das métricas quantitativas, imagens previstas foram geradas para uma avaliação visual do desempenho dos modelos. Elas permitiram analisar a precisão das previsões em exemplos específicos do conjunto de teste, proporcionando uma percepção intuitiva de como os modelos lidam com diferentes casos.

5 RESULTADOS E DISCUSSÕES

Nesta seção, são apresentados os resultados dos modelos de classificação de resíduos, avaliados por meio de métricas de desempenho, gráficos de treinamento e validação, matrizes de confusão e relatórios de classificação. Esses elementos permitem analisar a eficácia de cada modelo, destacando seus pontos fortes e limitações. O objetivo é identificar o modelo mais eficiente, considerando tanto métricas quantitativas quanto observações qualitativas.

FIGURA 39 – Resultados do treinamento do Modelo 0



Fonte: Autora

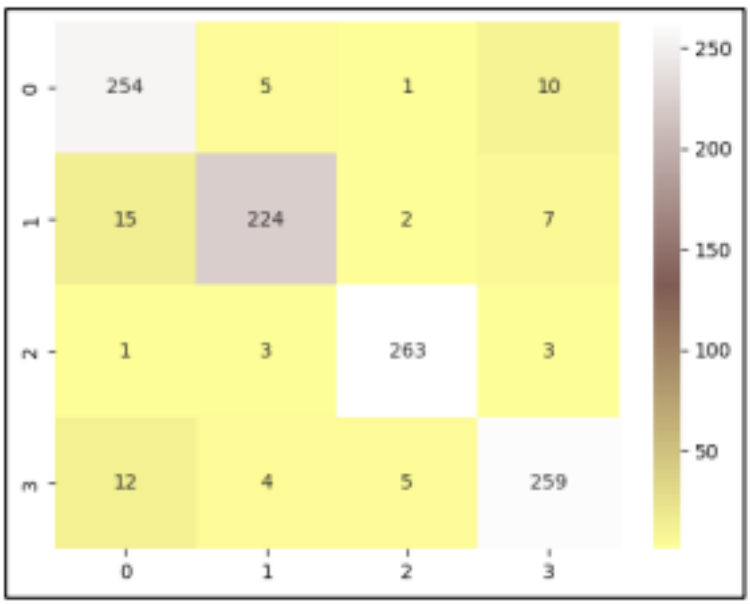
O **Modelo 0** apresentou resultados promissores, com gráficos de perda e acurácia que indicam um aprendizado consistente ao longo das épocas, evidenciando sua capacidade inicial de generalização.

A perda no conjunto de treinamento diminuiu rapidamente, atingindo valores baixos e estáveis, enquanto a perda de validação oscilou nas primeiras épocas antes de atingir níveis aceitáveis. Esse comportamento sugere que o modelo conseguiu generalizar razoavelmente bem, embora o início instável da validação aponte para uma sensibilidade a ruídos ou inconsistências nos dados de teste.

Com a acurácia de treinamento ultrapassando 97% e a validação estabilizando em torno de 93%, nota-se uma discrepância moderada que aponta

para um modelo bem estruturado, mas ainda com limitações na generalização para exemplos não vistos, sugerindo a necessidade de ajustes para otimizar seu desempenho.

FIGURA 40 – Matriz de confusão do Modelo 0



Fonte: Autora

A matriz de confusão mostra que o modelo teve dificuldade em diferenciar classes específicas, como Vidro (0) e Papel (3), além de confusões frequentes entre Metal (1) e Vidro (0). Esse padrão indica que o modelo não capturou plenamente as características distintivas dessas categorias, possivelmente devido à semelhança visual entre os resíduos ou limitações no conjunto de treinamento.

FIGURA 41 – Relatório de classificação do Modelo 0

```
---------------------- CLASSIFICATION REPORT ----------------------

              precision    recall  f1-score   support

           0       0.90      0.94      0.92       270
           1       0.95      0.90      0.93       248
           2       0.97      0.97      0.97       270
           3       0.93      0.93      0.93       280

    accuracy                           0.94      1068
   macro avg       0.94      0.94      0.94      1068
weighted avg       0.94      0.94      0.94      1068


Loss: 0.2242  |  Accuracy: 0.9363  |  Sensitivity: 0.9358  |  Specificity: 0.9787
```

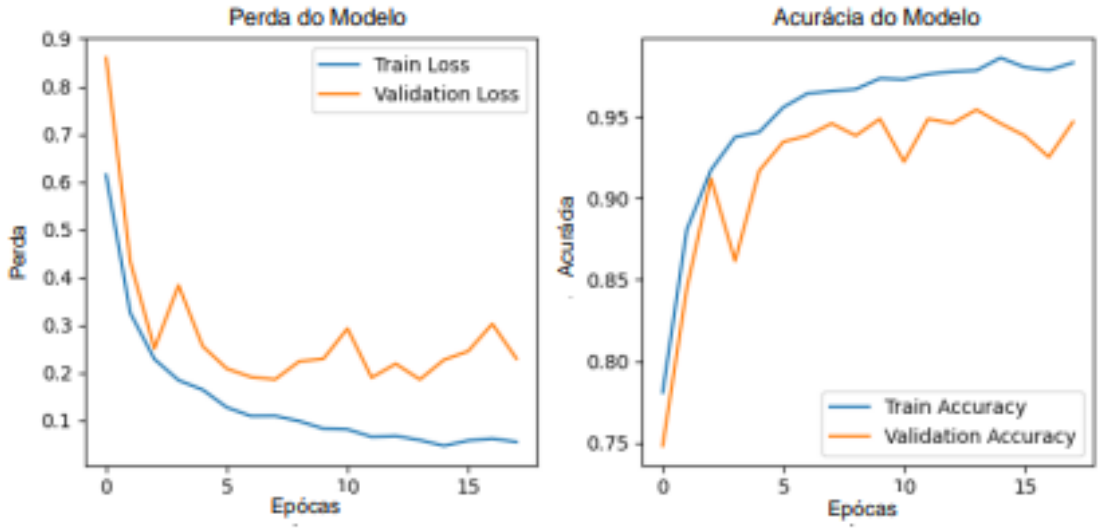
Fonte: Autora

O relatório de classificação aponta que o modelo teve os melhores desempenhos nas classes Plástico (2) e Papel (3), com F1-*scores* de 97% e 93%, respectivamente. Entretanto, a classe Vidro (0) apresentou *recall* inferior a 90%, indicando dificuldades em identificar corretamente todos os exemplos dessa categoria.

O Modelo 0 alcançou métricas satisfatórias, com acurácia de 93,63%, sensibilidade de 93,58% e especificidade de 97,87%, demonstrando seu potencial como uma base sólida. O aprendizado por transferência com a arquitetura MobileNetV2 e pesos pré-treinados no ImageNet forneceu uma estrutura robusta, mas a única camada densa de 64 neurônios limitou sua capacidade de aprender características mais complexas, contribuindo para confusões entre classes visualmente semelhantes.

O *Dropout* leve (0,08) ajudou a reduzir o risco de overfitting, mas mostrou-se insuficiente para lidar com ruídos em classes mais desafiadoras. Esses aspectos destacam a necessidade de refinamentos na arquitetura e no treinamento para melhorar a generalização e reduzir as confusões observadas.

FIGURA 42 – Resultados do treinamento do Modelo 1



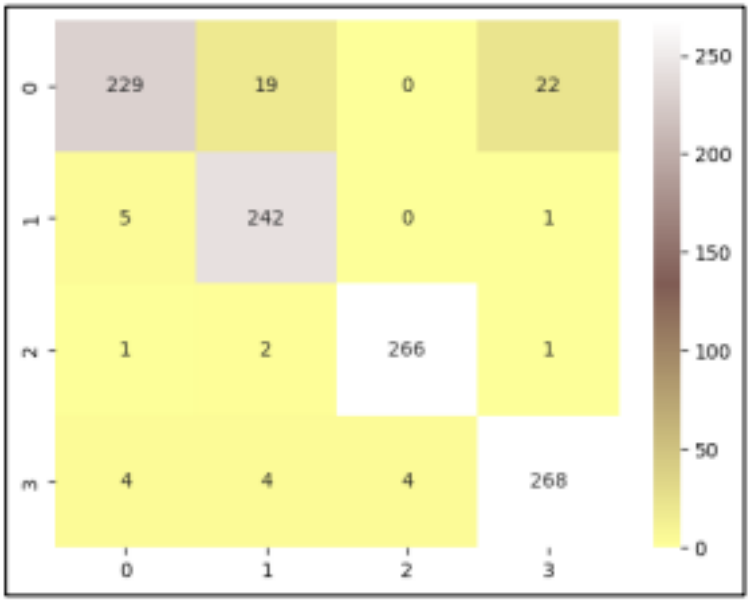
Fonte: Autora

O **Modelo 1** apresentou uma evolução clara em relação ao Modelo 0, com gráficos de perda e acurácia que evidenciam um aprendizado mais refinado e eficiente. A perda no conjunto de treinamento estabilizou-se rapidamente em

valores baixos, enquanto a perda de validação apresentou oscilações moderadas nas últimas épocas. Esse comportamento sugere que o modelo conseguiu aprender de forma consistente, embora ainda mostre sensibilidade a ruídos ou inconsistências nos dados de teste. De maneira geral, os níveis alcançados na validação refletem um aprendizado mais equilibrado e competitivo.

A acurácia de treinamento ultrapassou 95%, enquanto a validação estabilizou-se próxima a 94%, demonstrando uma redução da discrepância observada no Modelo 0. Esses resultados indicam um avanço na capacidade do modelo de generalizar para exemplos não vistos, embora haja espaço para aprimoramentos.

FIGURA 43 – Matriz de confusão do Modelo 1



Fonte: Autora

A matriz de confusão evidencia avanços no aprendizado do Modelo 1, com uma redução notável nas confusões observadas no Modelo 0. As classes Metal (1) e Plástico (2) demonstraram maior precisão, mas persistem dificuldades na diferenciação entre Vidro (0) e Metal (1), além de confusões pontuais envolvendo Vidro (0) e Papel (3). Embora menos frequentes, esses erros mostram que o modelo captou melhor as características discriminativas, mas ainda apresenta limitações em classes visualmente semelhantes, como Vidro.

FIGURA 44 – Relatório de classificação do Modelo 1

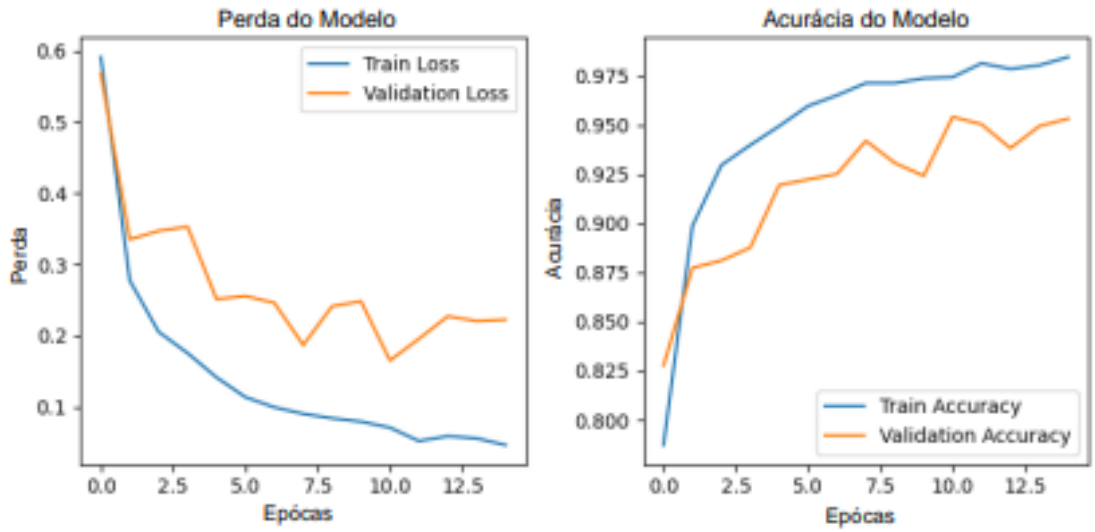
```
---------------------- CLASSIFICATION REPORT ----------------------

              precision    recall  f1-score   support

           0       0.96      0.85      0.90       270
           1       0.91      0.98      0.94       248
           2       0.99      0.99      0.99       270
           3       0.92      0.96      0.94       280

    accuracy                           0.94      1068
   macro avg       0.94      0.94      0.94      1068
weighted avg       0.94      0.94      0.94      1068


Loss: 0.2323  |  Accuracy: 0.9410  |  Sensitivity: 0.9416  |  Specificity: 0.9804
```

Fonte: Autora

Os resultados do relatório de classificação destacam que a classe Plástico (2) obteve o melhor desempenho, com um F1-*score* de 99%, enquanto a classe Vidro (0) apresentou menor *recall*, de 85%. Apesar das dificuldades na classificação de Vidro, as métricas globais refletem um modelo mais equilibrado, com acurácia de 94,10%, sensibilidade de 94,16% e especificidade de 98,04%.

O progresso do Modelo 1 em relação ao Modelo 0 é evidente, com redução significativa nas confusões e maior estabilidade no aprendizado. O aumento do *Dropout* para 0,2 favoreceu a estabilidade e ajudou a reduzir o overfitting. Contudo, esse ajuste pode ter comprometido o aprendizado detalhado, impactando o desempenho em classes desafiadoras, como Vidro.

FIGURA 45 – Resultados do treinamento do Modelo 2

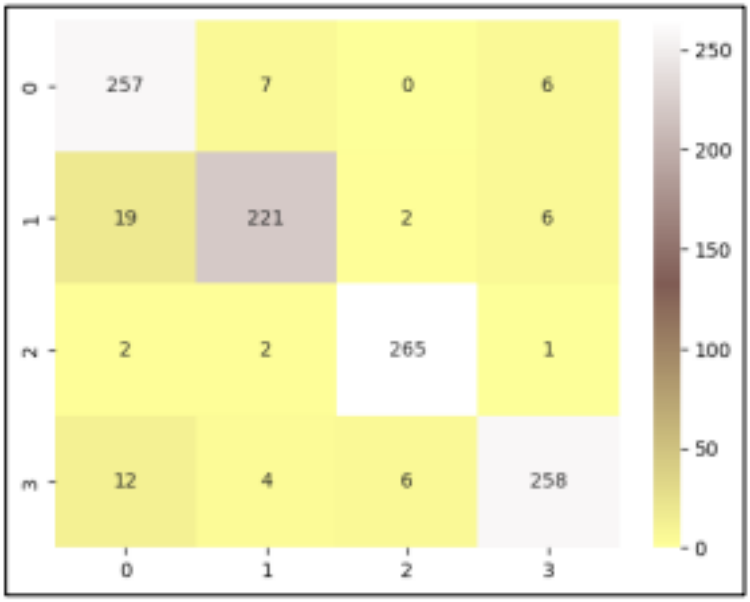


Fonte: Autora

O **Modelo 2** apresentou desempenho moderado, com gráficos de perda indicando uma diminuição consistente no treinamento, enquanto a validação exibiu oscilações ao longo das épocas antes de estabilizar em níveis aceitáveis. Esses resultados indicam que o modelo conseguiu aprender padrões relevantes nos dados, mas ainda apresenta espaço para refinamentos na arquitetura ou nos hiperparâmetros, como o ajuste da taxa de aprendizado e do número de neurônios, para melhorar sua capacidade de generalização.

A acurácia de treinamento ultrapassou 97%, enquanto a validação estabilizou-se em torno de 93%, indicando um equilíbrio razoável, embora com uma leve discrepância que não caracteriza um *overfitting* evidente.

FIGURA 46 – Matriz de confusão do Modelo 2



Fonte: Autora

A matriz de confusão confirma essas dificuldades, com a classe Vidro (0) acertando 257 amostras entre 270, mas apresentando 19 erros com Metal (1) e 6 com Papel (3). Já a classe Metal (1) obteve 221 acertos, enquanto Plástico (2) alcançou o melhor desempenho com 265 acertos. Apesar de alguns avanços pontuais, as confusões persistentes, especialmente envolvendo Vidro e Metal, indicam que o modelo não foi eficaz em capturar as características discriminativas necessárias para essas classes.

FIGURA 47 – Relatório de classificação do Modelo 2

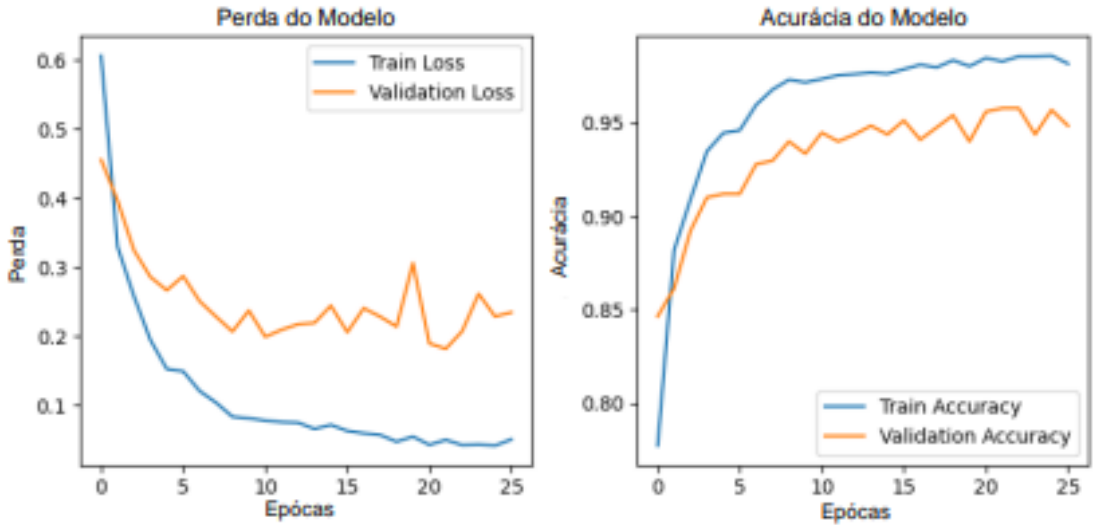
```
---------------------- CLASSIFICATION REPORT ----------------------

              precision    recall  f1-score   support

           0       0.89      0.95      0.92       270
           1       0.94      0.89      0.92       248
           2       0.97      0.98      0.98       270
           3       0.95      0.92      0.94       280

    accuracy                           0.94      1068
   macro avg       0.94      0.94      0.94      1068
weighted avg       0.94      0.94      0.94      1068


Loss: 0.2062  |  Accuracy: 0.9373  |  Sensitivity: 0.9365  |  Specificity: 0.9791
```

Fonte: Autora

O relatório de classificação indica um desempenho sólido, com F1-*scores* variando de 92% nas classes Vidro (0) e Metal (1) a 98% na classe Plástico (2). A acurácia geral foi de 93,73%, com sensibilidade de 93,65% e especificidade de 97,91%. Esses valores refletem uma evolução significativa em relação aos modelos anteriores, indicando que os ajustes estruturais trouxeram avanços no aprendizado das classes mais desafiadoras.

A inclusão de uma camada densa de 128 neurônios ampliou a capacidade de aprendizado do modelo, permitindo maior detalhamento na diferenciação entre classes. Apesar disso, algumas confusões persistem entre Vidro (0) e Papel (3), sugerindo refinamentos para melhorar a separação de classes semelhantes.

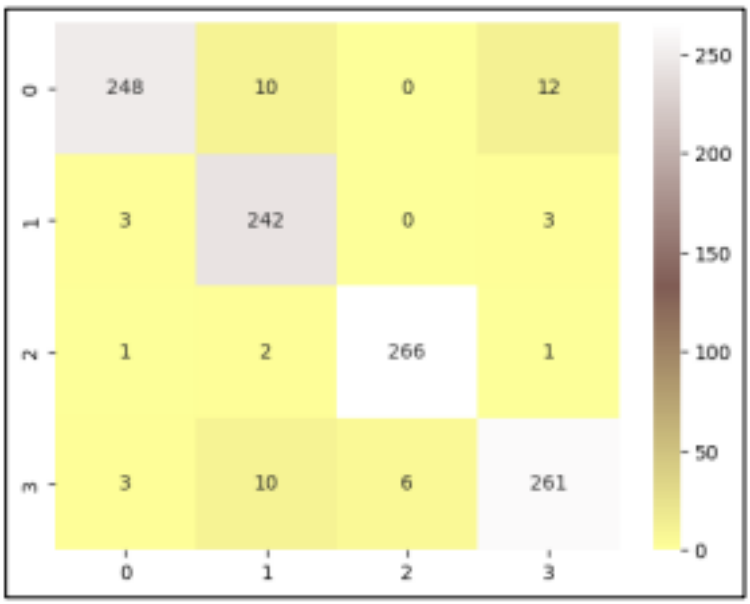
FIGURA 48 – Resultados do treinamento do Modelo 3



Fonte: Autora

O **Modelo 3** apresentou avanços significativos, destacando-se como a melhor versão entre as anteriores. O gráfico de perdas demonstra uma diminuição consistente no conjunto de treinamento, com valores baixos e estáveis ao final das épocas. A perda de validação apresentou leves oscilações, mas permaneceu em níveis baixos, indicando que o modelo conseguiu manter sua capacidade de generalização sem sinais evidentes de overfitting. O gráfico de acurácia mostra uma curva ascendente em ambos os conjuntos, com a acurácia de validação acompanhando de perto o treinamento, estabilizando em torno de 95%. Isso sugere que o modelo foi bem ajustado, apresentando um bom equilíbrio entre o aprendizado e a capacidade de generalizar para dados não vistos.

FIGURA 49 – Matriz de confusão do Modelo 3

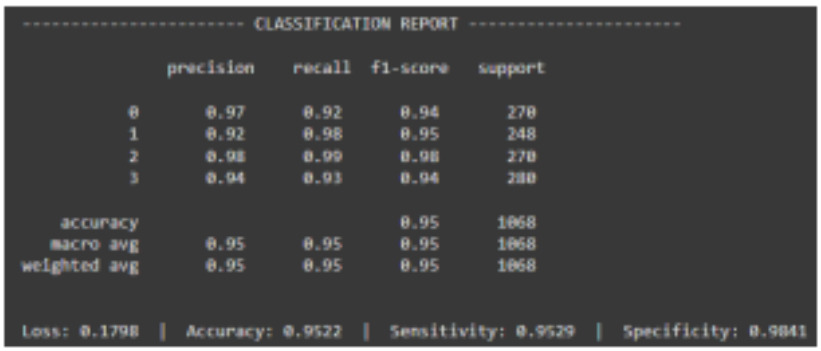


Fonte: Autora

A matriz de confusão destacou melhorias significativas na classificação em comparação aos modelos anteriores. A classe Vidro (0) teve 248 acertos em 270 amostras, com apenas 12 confusões com Papel (3) e 10 com Metal (1), refletindo um avanço em relação às confusões observadas nos modelos anteriores. Metal (1) e Plástico (2) mantiveram um desempenho elevado, com 242 e 266 acertos, respectivamente, demonstrando maior estabilidade no aprendizado dessas categorias. A classe Papel (3) também registrou progresso, com 261 acertos e confusões reduzidas.

Esses resultados indicam que o modelo conseguiu captar melhor as características discriminativas entre classes, especialmente em categorias anteriormente mais desafiadoras, como Vidro, embora ainda existam pequenos ajustes a serem feitos para alcançar maior precisão.

FIGURA 50 – Relatório de classificação do Modelo 3



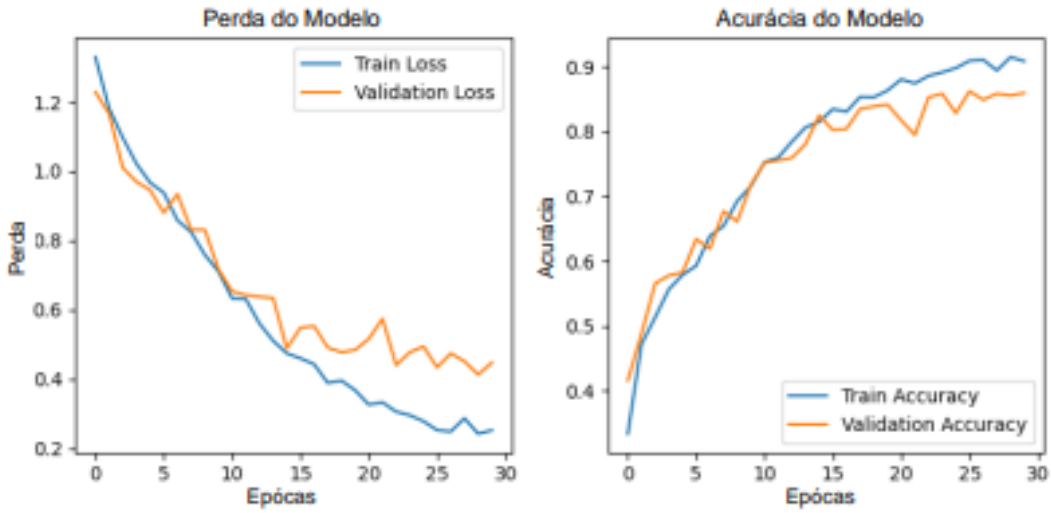
Fonte: Autora

O modelo alcançou métricas globais impressionantes, com acurácia de 95,22%, sensibilidade de 95,29% e especificidade de 98,41%. A classe Plástico (2) apresentou o melhor desempenho, com um F1-*score* de 98%, enquanto as classes Vidro (0) e Papel (3) registraram F1-*scores* de 94%. Esses resultados demonstram que o modelo foi capaz de capturar melhor as características discriminativas, reduzindo as confusões em classes visualmente semelhantes. No entanto, a classe Vidro ainda apresentou alguns desafios, exigindo ajustes mais finos.

A inclusão de uma segunda camada densa, com 32 neurônios, contribuiu para um aprendizado mais detalhado, equilibrando a capacidade do modelo e reduzindo os erros. Essa modificação favoreceu a estabilidade no treinamento e resultou em uma performance superior em comparação aos modelos anteriores, especialmente em termos de generalização.

Com métricas sólidas e uma redução clara nas confusões, o modelo demonstra estar bem mais ajustado para capturar padrões complexos e generalizar melhor, especialmente nas classes mais desafiadoras. Ainda assim, há espaço para refinamentos adicionais, visando alcançar uma separação ainda mais precisa entre classes com maior similaridade visual e melhorar o desempenho em classes mais problemáticas.

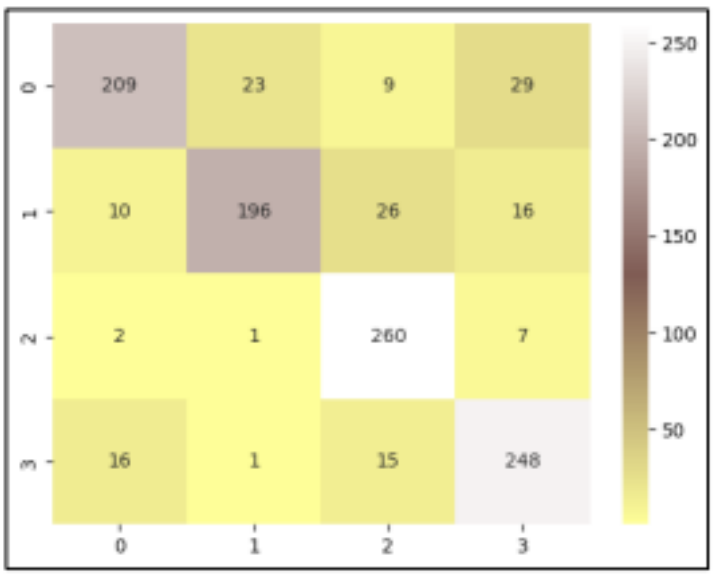
FIGURA 51 – Resultados do treinamento do Modelo 4



Fonte: Autora

O **Modelo 4** apresentou resultados consistentes, mas com limitações na generalização. O gráfico de perdas indica uma queda significativa no início do treinamento, estabilizando-se em níveis baixos ao longo das épocas. Já a perda de validação permaneceu moderada, com oscilações nas últimas épocas, sugerindo boa generalização apesar de desafios em cenários mais complexos. A acurácia de validação cresceu de forma constante, mas estabilizou em 85%, refletindo limitações na obtenção de valores mais altos.

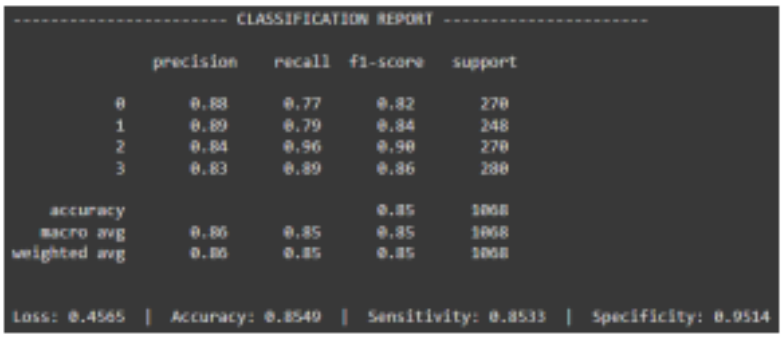
FIGURA 52 – Matriz de confusão do Modelo 4



Fonte: Autora

A matriz de confusão mostra um bom desempenho geral, com 209 acertos para Vidro (0) e 260 para Plástico (2). No entanto, houve confusões entre Vidro (0) e Papel (3) e entre Metal (1) e Vidro (0), sugerindo que as diferenças visuais entre essas classes podem ser sutis. As classes Plástico (2) e Papel (3) apresentaram os desempenhos mais consistentes, com menos confusões.

FIGURA 53 – Relatório de classificação do Modelo 4



Fonte: Autora

O relatório de classificação indica desempenho moderado, com F1-*scores* de 82% para Vidro (0) e 90% para Plástico (2). A acurácia geral foi de 85,49%, com sensibilidade de 85,33% e especificidade de 95,14%. Apesar de padrões satisfatórios em algumas classes, houve limitações em categorias desafiadoras, como Vidro e Metal, devido a dificuldades em captar características sutis.

A arquitetura de blocos convolucionais com quatro camadas, seguidas por duas camadas densas de 128 neurônios, ampliou sua capacidade de aprendizado, mas também aumentou a suscetibilidade a ruídos nos dados, justificando a menor sensibilidade em algumas classes.

TABELA 2 – Análise comparativa dos modelos

MODEL	ACCURACY	RECALL	SPECIFICITY	F1-SCORE			
				VIDRO	METAL	PLÁSTICO	PAPEL
Modelo 0	93.63	93,58	97,87	92	93	97	93
Modelo 1	94,10	94,16	98,04	90	94	99	94
Modelo 2	93,73	93,65	97,91	92	92	98	94
Modelo 3	95,22	95,29	98,41	94	95	98	94
Modelo 4	85,49	85,33	95,14	82	84	90	86

Fonte: Autora

Após os treinamentos e a validação dos resultados, observa-se que o Modelo 3 apresentou as melhores métricas de desempenho, destacando-se pela alta acurácia e sensibilidade. O Modelo 1 também se mostrou eficiente, com métricas consistentes e F1-scores elevados, especialmente na classe Plástico. O Modelo 4 apresentou desafios significativos em classes como Vidro e Metal, resultando em uma acurácia inferior. Por fim, o Modelo 2, embora tenha superado o Modelo 4 em algumas métricas, ainda mostrou desempenho abaixo dos demais modelos em termos de generalização.

6 CONCLUSÃO

Este trabalho explorou a aplicação de redes neurais convolucionais na classificação automatizada de resíduos sólidos urbanos, demonstrando a viabilidade e eficiência dessas tecnologias na gestão de resíduos. A metodologia abrangeu o uso de arquiteturas pré-treinadas e redes desenvolvidas do zero, com treinamento realizado em um banco de imagens categorizadas e o emprego de técnicas para equilibrar o aprendizado, como regularização por *Dropout* e ajustes progressivos em camadas densas.

Os resultados mostraram métricas satisfatórias, com destaque para o Modelo 3, que obteve acurácia de 95,22% e sensibilidade de 95,29%. Apesar disso, persistiram desafios na separação de classes com maior similaridade visual, como Vidro e Papel. Embora os resultados sejam promissores, limitações relacionadas à variabilidade dos dados e ao refinamento da arquitetura foram observadas, sugerindo caminhos para melhorias futuras, como ampliar a diversidade do conjunto de dados e ajustar hiperparâmetros para aumentar a robustez do sistema.

Ademais, trabalhos futuros podem explorar a implementação de sistemas robóticos para realizar, de forma automatizada e eficiente, a separação física dos materiais, ampliando a aplicabilidade prática da solução desenvolvida.

Ao combinar aprendizado de máquina e visão computacional, o trabalho ofereceu uma solução prática e eficaz para a gestão de resíduos. Além de otimizar a separação de materiais recicláveis, os resultados destacam o potencial das redes neurais em promover a sustentabilidade, reduzir impactos ambientais e incentivar práticas mais conscientes.

REFERÊNCIAS

ABRELPE – Associação Brasileira de Empresas de Limpeza Pública e Resíduos Especiais. **ISWA.** ABRELPE, 2023. Disponível em: < https://abrelpe.org.br/iswa/>. Acesso em: 10 abr. 2024.

ABREMA – Associação Brasileira de Resíduos e Meio Ambiente. **Panorama dos Resíduos Sólidos no Brasil 2023.** ABREMA, 2023. 54 p. Disponível em: <https://abrema.org.br/pdf/Panorama_2023_P1.pdf>. Acesso em: 26 fev. 2024.

BISHOP, Christopher M. **Pattern Recognition and Machine Learning**. New York: Springer, 2006. 758 p. Disponível em: <https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf>. Acesso em: 28 ago. 2024.

BRASIL. Ministério do Meio Ambiente. Sistema Nacional de Informações sobre a Gestão dos Resíduos Sólidos. **Resíduos de Estabelecimentos Comerciais e Prestadores de Serviços.** SINIR/MMA, 2020a. Disponível em: <https://sinir.gov.br/informacoes/tipos-de-residuos/residuos-de-estabelecimentos-comerciais-e-prestadores-de-servicos/>. Acesso em: 29 fev. 2024.

BRASIL. Ministério do Meio Ambiente. Sistema Nacional de Informações sobre a Gestão dos Resíduos Sólidos. **Resíduos Sólidos Urbanos.** SINIR/MMA, 2020b. Disponível em: <https://sinir.gov.br/informacoes/tipos-de-residuos/residuos-solidos-urbanos/>. Acesso em: 11 jul. 2024.

BRASIL. Ministério do Desenvolvimento Regional. Secretaria Nacional de Saneamento. **Diagnóstico Temático Manejo de Resíduos Sólidos Urbanos.** Brasília: SNS/MDR, 2022a. 51 p. Disponível em: <https://antigo.mdr.gov.br/images/stories/ArquivosSNSA/Arquivos_PDF/Snis/RESIDUOS_SOLIDOS/2020/3_DIAGNOSTICO_TEMATICO_INFRAESTRURA_PARA_OS_SERVICOS_RS_SNIS_SET_2022.pdf>. Acesso em: 05 jun. 2024.

BRASIL. Ministério do Desenvolvimento Regional. Secretaria Nacional de Saneamento. **Plano Nacional de Saneamento Básico - PLANSAB**. Brasília: SNS/MDR, 2019. 240 p. Disponível em: <https://www.gov.br/cidades/pt-br/acesso-

a-informacao/acoes-e-programas/saneamento/plano-nacional-de-saneamento-basico-plansab/arquivos/Versao_Conselhos_Resoluo_Alta__Capa_Atualizada.pdf>. Acesso em: 03 mar. 2024.

BRASIL. Ministério do Meio Ambiente. Secretaria de Qualidade Ambiental. **Plano Nacional de Resíduos Sólidos - PLANARES**. Brasília: SQA/MMA, 2022b. 209 p. Disponível em: <https://portal-api.sinir.gov.br/wp-content/uploads/2022/07/Planares-B.pdf>. Acesso em: 03 mar. 2024.

FILHO, Nilton Pinto Ribeiro. **Visão Computacional: Um Novo Campo de Pesquisa em Cognição Visual**. Brasília: Revista Psicologia: Teoria e Pesquisa, 2012. 138-150 p. Disponível em: <https://periodicos.unb.br/index.php/revistaptp/article/view/17018>. Acesso em: 25 jul. 2024.

GERON, Aurélien. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.** 2. ed. Sebastopol: O'Reilly Media, 2019. Disponível em: <http://14.139.161.31/OddSem-0822-1122/Hands-On_Machine_Learning_with_Scikit-Learn-Keras-and-TensorFlow-2nd-Edition-Aurelien-Geron.pdf>. Acesso em: 09 out. 2024.

HAYKIN, Simon. **Neural networks and learning machines.** 3. ed. Upper Saddle River, New Jersey: Pearson, 2009. 937 p. Disponível em: <https://github.com/xinlin192/DeepLearning/blob/master/Neural%20Networks%20and%20Learning%20Machines%20(3rd%20Edition).pdf>. Acesso em: 28 ago. 2024.

ISWA – International Solid Waste Association. **O Futuro do Setor de Gestão de Resíduos**. ISWA, 2022. 30 p. Disponível em: <https://www.iswa.org/wp-content/uploads/2022/09/ISWA-Future-of-the-Waste-Management-Sector-Portuguese.pdf>. Acesso em: 03 mar. 2024.

MARK, Lutz. **Learning Python.** 4. ed. Sebastopol: O'Reilly Media, 2009. Disponível em: <https://cfm.ehu.es/ricardo/docs/python/Learning_Python.pdf>. Acesso em: 09 out. 2024.

NETO, Euripedes Purcinio Fernandes. **Visão Computacional para Identificação de Cores em Tempo Real com OpenCV e Python**. Brasília: UniCEUB – Centro Universitário de Brasília, 2020. 17p. Disponível em: <https://repositorio.uniceub.br/jspui/bitstream/prefix/15103/1/Eur%C3%ADpedes%20Vis%C3%A3o%20Computacional%20VF%20PB.pdf>. Acesso em: 30 jul. 2024.

OLIVEIRA, Fabricio Fantin de; STOLL, Bruno Bastos. **Prova de Conceito (Poc) para detecção de objetos usando Visão Computacional em uma Coleta Seletiva**. Vitória: Multivix, 2023. 20p. Disponível em: <https://multivix.edu.br/wp-content/uploads/2022/03/prova-de-conceito-poc-para-deteccao-de-objetos-usando-visao-computacional-em-uma-coleta-seletiva.pdf>. Acesso em: 23 out. 2024.

PINA, Hudson Murilo Leal; MARTINS, Michael Bryan Miranda. **Utilização De bibliotecas de Visão Computacional e Aprendizado de Máquina na identificação de Resíduos Sólidos (Metal e Vidro) aplicado a Coleta Seletiva**. Anapólis: UniEVANGÉLICA – Centro Universitário de Anapólis, 2018. 57p. Disponível em: <http://repositorio.aee.edu.br/jspui/bitstream/aee/300/1/TCC2_2018_01_HudsonMuriloLealPina_MichaelBryanMirandaMartins.pdf>. Acesso em: 23 out. 2024.

PNUMA - Programa das Nações Unidas para o Meio Ambiente. **Panorama Global do Manejo de Resíduos em 2024**. PNUMA, 2024. Disponível em: <https://www.unep.org/pt-br/resources/panorama-global-do-manejo-de-residuos-em-2024>. Acesso em: 04 mar. 2024.

RUSSELL, Stuart; NORVIG, Peter. **Artificial Intelligence – A Modern Approach**. 4. ed. Pearson, 2021. 1127p. Disponível em: <https://dl.ebooksworld.ir/books/Artificial.Intelligence.A.Modern.Approach.4th.Edition.Peter.Norvig.%20Stuart.Russell.Pearson.9780134610993.EBooksWorld.ir.pdf>. Acesso em: 01 ago. 2024.

SZELISKI, Richard. **Computer Vision: Algorithms and Applications**. 2. ed. Springer, 2022. 1232 p. Disponível em: < https://szeliski.org/Book/>. Acesso em: 28. ago. 2024.

UNEP – United Nations Environment Programme. ISWA – International Solid Waste Association. **Global Waste Management Outlook 2024**. Nairóbi: UNEP, 2024. 116 p. Disponível em: <https://wedocs.unep.org/bitstream/handle/20.500.11822/44939/global_waste_management_outlook_2024.pdf?sequence=3>. Acesso em: 04 mar. 2024.

APÊNDICE A – Código em Python

```
# Setup
!pip install kaggle
import os
from google.colab import drive
import random
import numpy as np
import pandas as pd
import seaborn as sns
from skimage.transform import resize
import matplotlib.pyplot as plt
import imghdr
import matplotlib.image as mpimg
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.applications import MobileNetV2
from keras import models, layers
from keras.models import Sequential, Model
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization, Dropout, Flatten, Dense, Input, Activation
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
drive.mount('/content/drive')

# DataFrame
!unzip -o '/content/drive/MyDrive/kaggle/input/garbage-dataset.zip' -d '/content/drive/MyDrive/kaggle/input/garbage-dataset'
df = pd.DataFrame()
imgPaths = []
labels = []
for className in os.listdir(dataDir):
    classPath = os.path.join(dataDir,className)
    for img in os.listdir(classPath):
        imgPath = os.path.join(classPath,img)
        imgPaths.append(imgPath)
        labels.append(className)
df = pd.DataFrame({
    'imgPath':imgPaths,
    'label':labels
})
df = df.sample(frac=1).reset_index(drop=True)
df

# Global parameters
image_size = (224, 224)
input_shape = (224, 224, 3)
num_classes = 4
batch_size = 64
```

```
epochs = 50
seed = 42

# Data spliting
def DataFrameSpliting(df, train_ratio, val_ratio, test_ratio)
    trainDf, testDf = train_test_split(df, test_size=test_ratio, stratify=df['label'], random_state=seed)
    trainDf, valDf = train_test_split(trainDf, test_size=val_ratio / (train_ratio + val_ratio), stratify=trainDf['label'],
random_state=seed)
    trainDf = trainDf.sample(frac=1).reset_index(drop=True)
    valDf = valDf.sample(frac=1).reset_index(drop=True)
    testDf = testDf.sample(frac=1).reset_index(drop=True)
    return trainDf, valDf, testDf
trainDf, valDf, testDf = DataFrameSpliting(df, 0.70, 0.15, 0.15)
print("Training DataFrame = " + str(len(trainDf)) + " images")
print("Validation DataFrame = " + str(len(valDf)) + " images")
print("Testing DataFrame = " + str(len(testDf)) + " images")

# Data preprocessing
i = 0
imgPaths = df['imgPath']
plt.figure(figsize=(8, 5))
for imgPath in random.sample(list(imgPaths), len(imgPaths)):
    plt.subplot(2, 4, i + 1)
    img = mpimg.imread(imgPath)
    img = resize(img, image_size)
    plt.imshow(img)
    plt.title(imgPath.split('/')[-2])
    plt.axis("off")
    i += 1
    if i >= 8:
      break
plt.tight_layout()
plt.show()
total = 0
for category in os.listdir(dataDir):
    count = 0
    for image in os.listdir(os.path.join(dataDir, category)):
        count += 1
        total += 1
    print(str(category).title() + ": " + str(count) + " photos")
print(f"\nTotal: {total} images")
class_indices = {class_name: index for index, class_name in enumerate(os.listdir(dataDir))}
class_counts_dict = {class_name: 0 for class_name in os.listdir(dataDir)}
for category in os.listdir(str(dataDir)):
    class_counts_dict[category] = len(os.listdir(str(dataDir) + "/" + category))

plt.figure(figsize=(8, 4))
plt.bar(class_counts_dict.keys(), class_counts_dict.values(), color='skyblue')
plt.xticks(list(class_indices.values()), list(class_indices.keys()))
plt.title('Class Distribution in Training Dataset')
```

```
plt.xlabel('Classes')
plt.ylabel('Number of Images')
plt.show()

# Data augmentation
datagenTrain = ImageDataGenerator(
        rescale = 1./255,
        zoom_range = (1.0, 1.2),
        horizontal_flip = True,
        vertical_flip = True,
        rotation_range = 45,
)
trainGenerator = datagenTrain.flow_from_dataframe(
    trainDf,
    x_col = 'imgPath',
    y_col = 'label',
    target_size = image_size,
    batch_size = batch_size,
    class_mode = 'categorical'
)
datagenVal = ImageDataGenerator(rescale=1./255)
valGenerator = datagenVal.flow_from_dataframe(
    valDf,
    x_col = 'imgPath',
    y_col = 'label',
    target_size = image_size,
    batch_size = 8,
    class_mode = 'categorical',
    shuffle = False
)
datagenTest = ImageDataGenerator(rescale=1./255)
testGenerator = datagenTest.flow_from_dataframe(
    testDf,
    x_col = 'imgPath',
    y_col = 'label',
    target_size = image_size,
    batch_size = 8,
    class_mode = 'categorical',
    shuffle = False
)

# Model0
Model0 = Sequential([
    Input(shape=input_shape),
    MobileNetV2(weights='imagenet',include_top=False,input_shape=input_shape),
    Flatten(),
    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.08),
    Dense(num_classes, activation='softmax')
```

```
])
preTrainedModel0 = Model0.layers[0]
for layer in preTrainedModel0.layers[:-4]:
    layer.trainable = False
Model0.compile(optimizer='adam',
            loss='categorical_crossentropy',
            metrics=['accuracy']
            )
Model0.summary()

# Training0
history0 = Model0.fit(trainGenerator,
                validation_data = valGenerator,
                epochs = epochs,
                verbose = 1,
                callbacks = [EarlyStopping(
                            patience = 4,
                            monitor = 'val_accuracy',
                            restore_best_weights = True)
                ])
fig, axs = plt.subplots(1, 2, figsize=(8, 4))
axs[0].plot(history0.history['loss'], label='Train Loss')
axs[0].plot(history0.history['val_loss'], label='Validation Loss')
axs[0].set_title('Model Loss')
axs[0].set_ylabel('Loss')
axs[0].set_xlabel('Epoch')
axs[0].legend(loc='upper right')
axs[1].plot(history0.history['accuracy'], label='Train Accuracy')
axs[1].plot(history0.history['val_accuracy'], label='Validation Accuracy')
axs[1].set_title('Model Accuracy')
axs[1].set_ylabel('Accuracy')
axs[1].set_xlabel('Epoch')
axs[1].legend(loc='lower right')
plt.tight_layout()
plt.show()

# Model1
Model1 = Sequential([
    Input(shape=input_shape),
    MobileNetV2(weights='imagenet',include_top=False,input_shape=input_shape),
    Flatten(),
    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),
    Dense(num_classes, activation='softmax')
])
preTrainedModel1 = Model1.layers[0]
for layer in preTrainedModel1.layers[:-4]:
    layer.trainable = False
Model1.compile(optimizer='adam',
```

```
        loss='categorical_crossentropy',
        metrics=['accuracy']
        )
Model1.summary()

# Training1
history1 = Model1.fit(trainGenerator,
            validation_data = valGenerator,
            epochs = epochs,
            #batch_size=64,
            verbose = 1,
            callbacks = [EarlyStopping(
                        patience = 4,
                        monitor = 'val_accuracy',
                        restore_best_weights = True)
            ])
fig, axs = plt.subplots(1, 2, figsize=(8, 4))
axs[0].plot(history1.history['loss'], label='Train Loss')
axs[0].plot(history1.history['val_loss'], label='Validation Loss')
axs[0].set_title('Model Loss')
axs[0].set_ylabel('Loss')
axs[0].set_xlabel('Epoch')
axs[0].legend(loc='upper right')
axs[1].plot(history1.history['accuracy'], label='Train Accuracy')
axs[1].plot(history1.history['val_accuracy'], label='Validation Accuracy')
axs[1].set_title('Model Accuracy')
axs[1].set_ylabel('Accuracy')
axs[1].set_xlabel('Epoch')
axs[1].legend(loc='lower right')
plt.tight_layout()
plt.show()

# Model2
Model2 = Sequential([
    Input(shape=input_shape),
    MobileNetV2(weights='imagenet',include_top=False,input_shape=input_shape),
    Flatten(),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.08),
    Dense(num_classes, activation='softmax')
])
preTrainedModel2 = Model2.layers[0]
for layer in preTrainedModel2.layers[:-4]:
    layer.trainable = False
Model2.compile(optimizer='adam',
        loss='categorical_crossentropy',
        metrics=['accuracy']
        )
Model2.summary()
```

```
# Training2
history2 = Model2.fit(trainGenerator,
                validation_data = valGenerator,
                epochs = epochs,
                #batch_size=64,   # we define it above inside trainGenerator
                verbose = 1,
                callbacks = [EarlyStopping(
                                patience = 4,
                                monitor = 'val_accuracy',
                                restore_best_weights = True)
                ])
fig, axs = plt.subplots(1, 2, figsize=(8, 4))
axs[0].plot(history2.history['loss'], label='Train Loss')
axs[0].plot(history2.history['val_loss'], label='Validation Loss')
axs[0].set_title('Model Loss')
axs[0].set_ylabel('Loss')
axs[0].set_xlabel('Epoch')
axs[0].legend(loc='upper right')
axs[1].plot(history2.history['accuracy'], label='Train Accuracy')
axs[1].plot(history2.history['val_accuracy'], label='Validation Accuracy')
axs[1].set_title('Model Accuracy')
axs[1].set_ylabel('Accuracy')
axs[1].set_xlabel('Epoch')
axs[1].legend(loc='lower right')
plt.tight_layout()
plt.show()

# Model3
Model3 = Sequential([
   Input(shape=input_shape),
   MobileNetV2(weights='imagenet',include_top=False,input_shape=input_shape),
   Flatten(),
   Dense(64, activation='relu'),
   Dense(32, activation='relu'),
   BatchNormalization(),
   Dropout(0.08),
   Dense(num_classes, activation='softmax')
])
preTrainedModel3 = Model3.layers[0]
for layer in preTrainedModel3.layers[:-4]:
   layer.trainable = False
Model3.compile(optimizer='adam',
          loss='categorical_crossentropy',
          metrics=['accuracy']
          )
Model3.summary()

# Training3
history3 = Model3.fit(trainGenerator,
```

```
                validation_data = valGenerator,
                epochs = epochs,
                #batch_size=64,   # we define it above inside trainGenerator
                verbose = 1,
                callbacks = [EarlyStopping(
                                patience = 4,
                                monitor = 'val_accuracy',
                                restore_best_weights = True)
                ])
fig, axs = plt.subplots(1, 2, figsize=(8, 4))
axs[0].plot(history3.history['loss'], label='Train Loss')
axs[0].plot(history3.history['val_loss'], label='Validation Loss')
axs[0].set_title('Model Loss')
axs[0].set_ylabel('Loss')
axs[0].set_xlabel('Epoch')
axs[0].legend(loc='upper right')
axs[1].plot(history3.history['accuracy'], label='Train Accuracy')
axs[1].plot(history3.history['val_accuracy'], label='Validation Accuracy')
axs[1].set_title('Model Accuracy')
axs[1].set_ylabel('Accuracy')
axs[1].set_xlabel('Epoch')
axs[1].legend(loc='lower right')
plt.tight_layout()
plt.show()


# Model4
Model4 = Sequential([
    Input(shape=input_shape),
    Conv2D(32, 3),
    Activation(activation="relu"),
    MaxPooling2D(),
    Conv2D(64, 3),
    Activation(activation="relu"),
    MaxPooling2D(),
    Conv2D(128, 3),
    Activation(activation="relu"),
    MaxPooling2D(),
    Conv2D(128, 3),
    Activation(activation="relu"),
    MaxPooling2D(),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(num_classes, activation='softmax')
])
optimizer = keras.optimizers.Adam(learning_rate=0.0007)
Model4.compile(optimizer=optimizer,
                loss='categorical_crossentropy',
```

```
            metrics=['accuracy'])
Model4.summary()

# Training4
history4 = Model4.fit(trainGenerator,
                validation_data = valGenerator,
                epochs = epochs,
                verbose = 1,
                callbacks = [EarlyStopping(
                            patience = 4,
                            monitor = 'val_accuracy',
                            restore_best_weights = True)
                ])
fig, axs = plt.subplots(1, 2, figsize=(8, 4))
axs[0].plot(history4.history['loss'], label='Train Loss')
axs[0].plot(history4.history['val_loss'], label='Validation Loss')
axs[0].set_title('Model Loss')
axs[0].set_ylabel('Loss')
axs[0].set_xlabel('Epoch')
axs[0].legend(loc='upper right')
axs[1].plot(history4.history['accuracy'], label='Train Accuracy')
axs[1].plot(history4.history['val_accuracy'], label='Validation Accuracy')
axs[1].set_title('Model Accuracy')
axs[1].set_ylabel('Accuracy')
axs[1].set_xlabel('Epoch')
axs[1].legend(loc='lower right')
plt.tight_layout()
plt.show()

# Predicted Images
class_labels = list(testGenerator.class_indices.keys())
def predicted_images(model, model_name):
    images = []
    predicted_labels = []
    true_labels = []
    for i in range(len(testGenerator)):
        img_batch, true_labels_batch = next(testGenerator)
        true_class_idx = np.argmax(true_labels_batch[0])
        prediction = model.predict(img_batch, verbose = 0)
        predicted_class_idx = np.argmax(prediction[0])
        predicted_class = class_labels[predicted_class_idx]
        true_class = class_labels[true_class_idx]
        images.append(np.squeeze(img_batch[0]))
        predicted_labels.append(predicted_class)
        true_labels.append(true_class)
        if i >= 12:  # For example, only plot 12 images
            break
    print(f"\n\n------ PREDICTED IMAGES - {model_name} ------\n\n")
    fig, axs = plt.subplots(2, 4, figsize=(8, 6))
    axs = axs.flatten()
```

```
    for ax,img , pred , true in zip(axs, images, predicted_labels, true_labels):
        img = resize(img, image_size)
        ax.imshow(img)
        ax.set_title(f"Predicted: {pred}\nTrue: {true}")
        ax.axis('off')
    plt.tight_layout()
    return plt.show()
predicted_images(Model0, "Model 0")
predicted_images(Model1, "Model 1")
predicted_images(Model2, "Model 2")
predicted_images(Model3, "Model 3")
predicted_images(Model4, "Model 4")

# Evaluation
def calculate_evaluation(model, model_name):
    loss, accuracy = model.evaluate(testGenerator, verbose = 0)
    predictions = model.predict(testGenerator, verbose = 0)
    true_labels = testGenerator.classes
    true_labels[:10]
    predicted_labels = predictions.argmax(axis=-1)
    predicted_labels[:10]

    print(f"_______ EVALUATION - {model_name} _______\n")
    print(f"------ CONFUSION MATRIX ------\n")
    cm = confusion_matrix(true_labels, predicted_labels)
    sns.heatmap(cm, center = True,cmap='terrain',annot=True ,fmt='.5g')
    plt.show()
    ClassificationReport = classification_report(true_labels, predicted_labels)
    print(f"\n------ CLASSIFICATION REPORT ------\n\n", ClassificationReport)
    FP = cm.sum(axis=0) - np.diag(cm)
    FN = cm.sum(axis=1) - np.diag(cm)
    TP = np.diag(cm)
    TN = cm.sum() - (FP + FN + TP)
    sensitivity = np.mean(TP / (TP + FN))
    specificity = np.mean(TN / (TN + FP))
    print(f"\nLoss: {loss:.4f}  |  Accuracy: {accuracy:.4f}")
    print(f"\nSensitivity: {sensitivity:.4f}  |  Specificity: {specificity:.4f}\n")
calculate_evaluation(Model0, "Model 0")
calculate_evaluation(Model1, "Model 1")
calculate_evaluation(Model2, "Model 2")
calculate_evaluation(Model3, "Model 3")
calculate_evaluation(Model4, "Model 4")
```