UNIVERSIDADE FEDERAL DE UBERLÂNDIA - UFU
FACULDADE DE ENGENHARIA QUÍMICA - FEQUI

JANDERSON LUCAS RIBEIRO

VISÃO COMPUTACIONAL NA ENGENHARIA QUÍMICA: REVISÃO TEÓRICA E IMPLEMENTAÇÃO DE CNNS PARA INSPEÇÃO DE PEÇAS METÁLICAS

UBERLÂNDIA
2025

JANDERSON LUCAS RIBEIRO

VISÃO COMPUTACIONAL NA ENGENHARIA QUÍMICA: REVISÃO TEÓRICA E IMPLEMENTAÇÃO DE CNNS PARA INSPEÇÃO DE PEÇAS METÁLICAS

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia Química (FEQUI) da Universidade Federal de Uberlândia para obtenção de título de Bacharelado em Engenharia Química.

Orientador: Prof. Dr. Sarah Arvelos Altino

UBERLÂNDIA

2025

RESUMO

A crescente demanda por eficiência e automação na indústria tem impulsionado o uso de inteligência artificial. Entre essas tecnologias, a visão computacional, apoiada por redes neurais convolucionais (CNNs), tem se destacado na detecção de defeitos em processos industriais e a aplicação dessas técnicas pode reduzir custos operacionais e aumentar a confiabilidade dos produtos finais. Assim, este trabalho investigou a implementação de uma CNN para a identificação de defeitos em peças metálicas, avaliando sua eficácia e aplicabilidade no contexto industrial. O estudo teve como objetivo principal desenvolver e avaliar um modelo de CNN para a classificação automática de peças defeituosas e normais. Para isso, foram definidos quatro cenários computacionais, variando a quantidade de camadas densas e o número de épocas de treinamento. Os modelos foram treinados e testados utilizando um conjunto de dados disponibilizado na plataforma Kaggle, contendo imagens de peças metálicas com e sem defeitos. A performance dos modelos foi analisada por meio de métricas como acurácia e matriz de confusão, complementadas por testes estatísticos, como ANOVA e Tukey. Os resultados indicaram que o modelo com duas camadas densas e dez épocas de treinamento apresentou o melhor desempenho, destacando-se principalmente pela menor incidência de falsos positivos, o que o torna mais confiável para aplicação industrial. Dessa forma, conclui-se que a aplicação das redes neurais convolucionais tem potencial de aplicação para inspeção de qualidade, proporcionando uma boa alternativa aos métodos tradicionais de verificação visual. Trabalhos futuros poderão explorar o ajuste fino de hiperparâmetros, o uso de bases de dados maiores e a aplicação de aprendizado por transferência para aprimorar os resultados.

Palavras-chaves: Visão computacional; Redes neurais convolucionais; Inteligência artificial.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A crescente demanda por qualidade e eficiência nos processos industriais impulsionou a adoção de tecnologias baseadas em inteligência artificial (IA). Entre essas tecnologias, a visão computacional vem ganhando destaque por sua capacidade de interpretar imagens e automatizar tarefas anteriormente dependentes da análise humana. Esse avanço tem sido particularmente relevante na manufatura e na engenharia química, onde a identificação de defeitos em produtos desempenha um papel fundamental na garantia da qualidade LeCun, Bengio e Hinton (2015). A utilização de redes neurais convolucionais (CNNs) na detecção de falhas representa um avanço significativo na área, permitindo a otimização dos processos produtivos e a redução de perdas associadas a produtos defeituosos (GOODFELLOW; BENGIO; COURVILLE, 2016).

A aplicação de inteligência artificial na indústria química não é apenas uma tendência tecnológica, mas uma necessidade crescente diante da complexidade dos processos produtivos. Métodos tradicionais de inspeção baseados em análise visual manual podem ser demorados e sujeitos a falhas humanas. Nesse contexto, as CNNs surgem como uma solução promissora, pois podem ser treinadas para reconhecer padrões em imagens de maneira automatizada e com alta precisão. No entanto, apesar dos avanços, muitos profissionais da área ainda desconhecem ou subestimam o potencial dessas ferramentas para a melhoria da qualidade e eficiência operacional.

Diante desse cenário, este trabalho busca explorar e evidenciar a aplicação prática das CNNs na engenharia química por meio de uma revisão teórica abrangente e um estudo de caso computacional. Inicialmente, são apresentados diversos estudos que demonstram como a visão computacional tem sido empregada na otimização de processos químicos, contribuindo para a automação e eficiência em diferentes setores industriais. Em seguida, para consolidar essa abordagem, é desenvolvido um modelo de CNN aplicado à inspeção de qualidade de peças metálicas de fundição, exemplificando como essa tecnologia pode ser implementada na prática e analisando seu desempenho.

A pesquisa proposta busca responder à seguinte questão: "É possível utilizar redes neurais convolucionais para identificar, com alta precisão, defeitos em peças metálicas de fundição?"Para isso, foi desenvolvido um estudo computacional que inclui a construção e o treinamento de modelos de CNNs para análise de imagens. A implementação desse modelo visa demonstrar que as CNNs não são apenas ferramentas restritas ao campo da ciência da computação, mas podem ser adotadas por engenheiros químicos e outros profissionais para resolver problemas industriais reais.

A escolha do tema justifica-se pela crescente necessidade de otimização dos processos produtivos na indústria metalúrgica e pela busca por métodos mais eficazes de controle de qualidade. A automação da inspeção por meio de visão computacional pode reduzir custos operacionais, minimizar desperdícios e melhorar a confiabilidade dos produtos finais (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Além disso, a integração de técnicas de aprendizado profundo com sistemas industriais representa um avanço na implementação da Indústria 4.0, alinhando-se às tendências de digitalização e automação dos processos produtivos.

Com essa abordagem, este trabalho busca não apenas reforçar a relevância da visão computacional na engenharia química, mas também incentivar sua adoção como uma ferramenta acessível para profissionais da área. Ao demonstrar, por meio de um estudo de caso, que as CNNs podem ser aplicadas de maneira eficaz na inspeção de qualidade industrial, espera-se contribuir para a disseminação desse conhecimento e estimular futuras aplicações dessa tecnologia em diferentes segmentos da engenharia química.

1.1 Objetivo Geral

Este estudo tem como objetivo geral desenvolver e avaliar um modelo de rede neural convolucional para a identificação automática de defeitos em peças metálicas de fundição, utilizando imagens e técnicas de aprendizado profundo.

1.2 Objetivo Específico

1. Revisar os conceitos fundamentais de redes neurais convolucionais e sua aplicação em visão computacional;

2. Implementar e treinar modelos de CNNs para a classificação de peças defeituosas e normais;

3. Avaliar o desempenho dos modelos utilizando métricas como acurácia e matriz de confusão;

4. Comparar diferentes configurações de rede neural para identificar a mais eficiente na tarefa proposta.

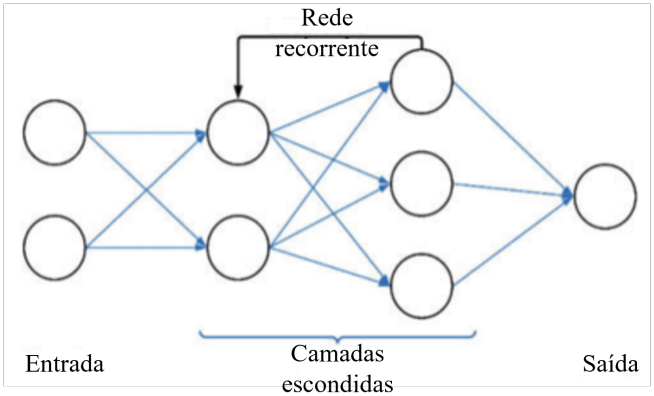
2 FUNDAMENTAÇÃO TEÓRICA

Nas últimas décadas, o aumento significativo da complexidade dos problemas computacionais, aliado ao volume crescente de dados gerados em diversos setores da Indústria 4.0, revelou a necessidade de ferramentas computacionais mais sofisticadas e autônomas, capazes de otimizar processos e reduzir a dependência de intervenção humana. Nesse cenário, as Redes Neurais Artificiais (RNAs) despontaram como uma alternativa promissora, demonstrando aplicabilidade prática em uma variedade de contextos industriais e oferecendo soluções robustas para a análise e o controle de dados em larga escala (WU et al., 2023).

As redes neurais artificiais (RNAs) são modelos computacionais que simulam vagamente o funcionamento do sistema nervoso dos seres vivos, estruturando-se como sistemas distribuídos compostos por neurônios artificiais interconectados por sinapses artificiais, representadas matematicamente por vetores ou matrizes de pesos sinápticos. Essas redes possuem características notáveis, incluindo a capacidade de aprendizado e adaptação, habilidade de generalizar a partir de exemplos, tolerância a falhas e armazenamento de informações de maneira distribuída, o que as torna ferramentas eficazes para diversas aplicações na inteligência artificial e, consequentemente, na indústria (WU et al., 2023).

Elas são compostas composta por três camadas principais: camada de entrada, camadas ocultas e camada de saída, conforme esquema apresentado na Figura 1. A camada de entrada recebe os dados brutos do ambiente externo, como características específicas de imagens. Em seguida, as camadas ocultas, formadas por neurônios interconectados, processam essas informações, extraindo padrões complexos e identificando relações relevantes. Por fim, a camada de saída apresenta o resultado final, seja uma classificação ou uma previsão com base no aprendizado desenvolvido pelas camadas ocultas (SOUSA, 2022).

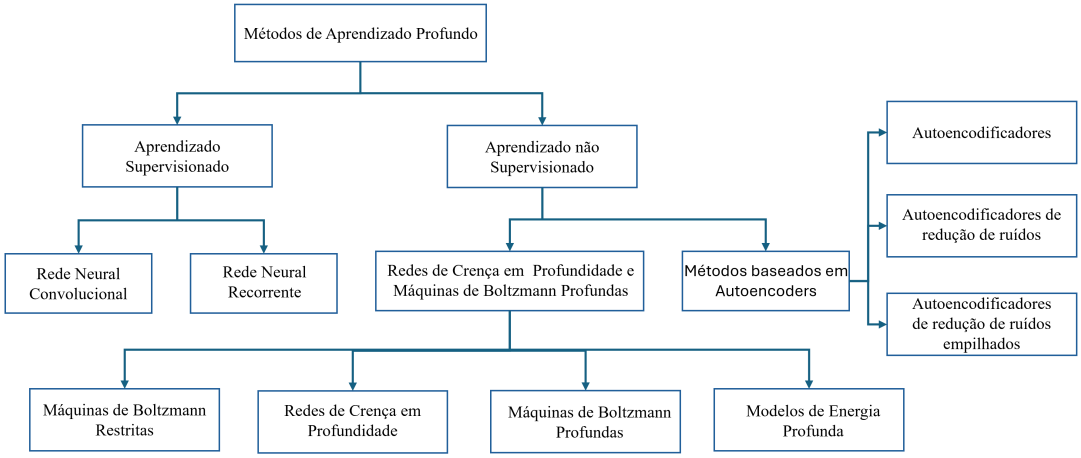
Figura 1 – Estrutura de uma rede neural artificial



Fonte: Adaptado de Ganesh et al. (2024).

A história das Redes Neurais Artificiais (RNAs) começou em 1943, com o modelo de neurônio artificial de McCulloch e Pitts, seguido pela regra de aprendizagem de Hebb em 1949, que fundamentou a ideia de aprendizado em redes. Nos anos 1980, o desenvolvimento do algoritmo de retropropagação e contribuições de Hopfield e LeCun revitalizaram o campo, permitindo o aprendizado em redes multicamadas. Com o avanço em hardware e técnicas, como as redes neurais convolucionais (CNNs), *Deep Learning* emergiu como um ramo de *Machine Learning*, utilizando arquiteturas profundas para resolver problemas complexos, como visão computacional, com desempenho superior às redes tradicionais (GANESH et al., 2024). A Figura 2 mostra os diferentes métodos de *Deep Learning*. Esse trabalho irá focar nas Redes Neurais Convolucionais.

Figura 2 – *Deep Learning* e seus algorítmos de aprendizagem supervisionada e não supervisionada



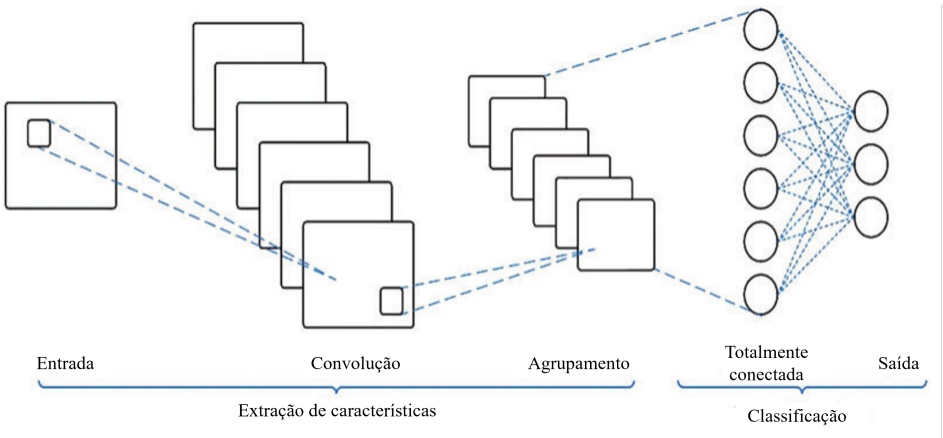
Fonte: Adaptado de Ganesh et al. (2024).

2.1 Redes Neurais Convolucionais

As redes neurais convolucionais (CNNs) são baseados em redes neurais de múltiplas camadas, projetados para identificar, reconhecer e classificar objetos, além de realizar tarefas como detecção e segmentação em imagens. Elas possuem a capacidade de aprender diretamente a partir dos dados de entrada. Essas redes têm sido amplamente aplicadas em diversas áreas, como reconhecimento visual, análise de imagens médicas, segmentação de imagens e processamento de linguagem natural (CHEN et al., 2021).

As redes neurais convolucionais (CNNs) diferem das redes neurais tradicionais principalmente pela forma como processam os dados e extraem características. Enquanto as redes tradicionais tratam todas as conexões entre neurônios de forma densa, as CNNs utilizam camadas convolucionais que compartilham pesos para realizar a extração das características, reduzindo significativamente os requisitos de memória. Além disso, nas CNNs, as camadas convolucionais focam na identificação de padrões relevantes, enquanto as camadas totalmente conectadas são responsáveis pela classificação. Toda essa estrutura está representada na Figura 3, e permite maior eficiência e desempenho no processamento de imagens, em comparação com redes tradicionais (BHARADIYA, 2023).

Figura 3 – Arquitetura da Rede Neural Convolucional



Fonte: Adaptado de Ganesh et al. (2024).

Elas destacam-se por quatro razões principais que reforçam sua importância em tarefas de processamento de imagens: invariância à translação, eficiência de dados, aprendizado por transferência e escalabilidade. A invariância à translação permite que reconheçam padrões independentemente da sua posição ou orientação, tornando-as robustas em cenários reais. Além disso, CNNs suportam aprendizado por transferência, reutilizando conhecimento de tarefas

anteriores por meio de modelos pré-treinados, o que reduz a necessidade de grandes volumes de dados e melhora o desempenho. Por fim, sua escalabilidade permite adaptação a diferentes níveis de complexidade, desde classificação de imagens até detecção e segmentação de objetos (BHARADIYA, 2023).

Um conceito fundamental para as redes neurais convolucionais são épocas de treinamento ou simplesmente épocas. A configuração desse parâmetro impacta nos requisitos de memória e apredizagem do modelo. Desse modo, a definição do número de épocas deve equilibrar a necessidade de aprendizado da rede sem comprometer a eficiência computacional, garantindo um treinamento adequado dentro das limitações de processamento disponíveis (AGGARWAL, 2018).

A arquitetura de uma CNN é composta por quatro estruturas: camada de convolução, camada de pooling, função de ativação e camada totalmente conectada.

2.1.1 Camada Convolucional

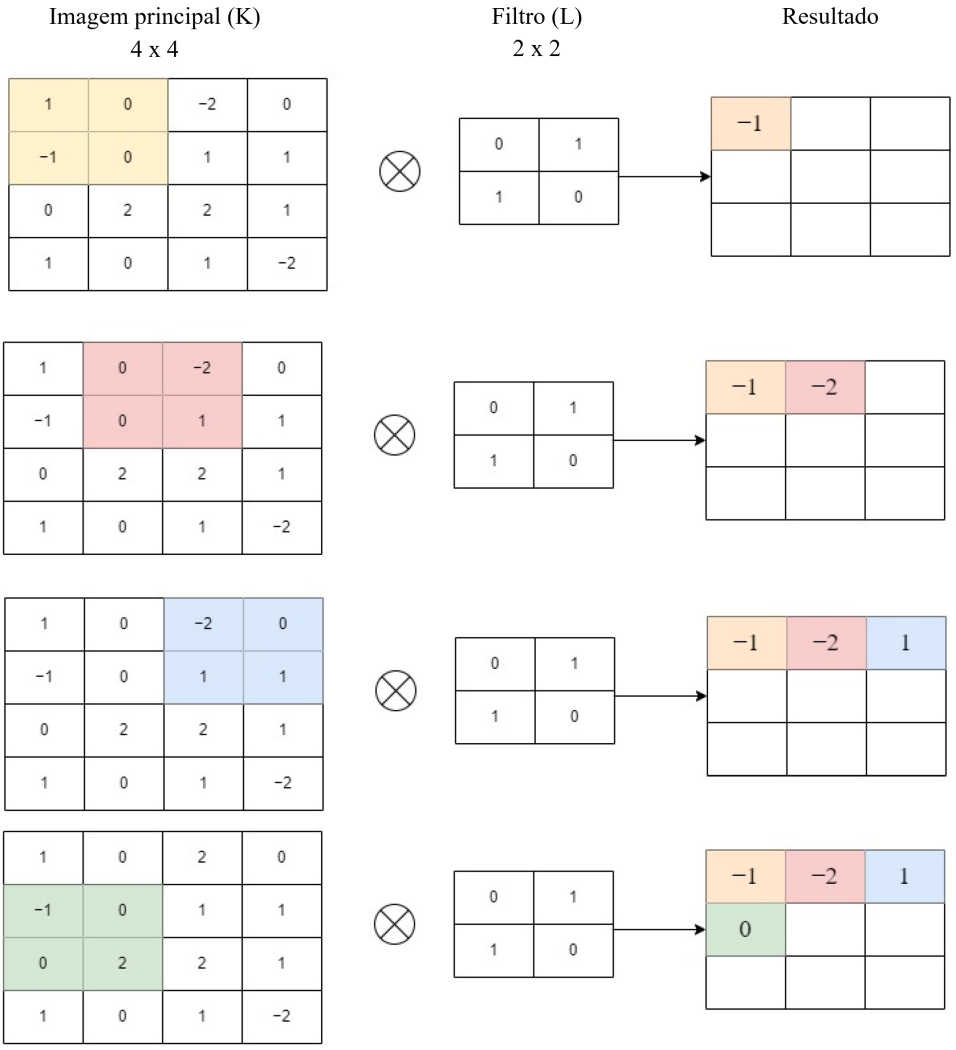
A camada convolucional é considerada o componente central de uma CNN, sendo responsável pela extração de características dos dados de entrada. Essa camada utiliza filtros, também chamados de kernels, que percorrem sequencialmente pequenas regiões do dado de entrada, realizando operações de convolução. Em cada interação, o filtro calcula o produto entre seus elementos e os elementos correspondentes da entrada, somando os resultados para gerar uma saída específica. O conjunto dessas saídas forma o chamado mapa de características, que representa padrões detectados nos dados (ZHAO et al., 2024).

Cada kernel é uma matriz composta por valores inteiros inicializados aleatoriamente, cujos pesos são ajustados durante o treinamento para extrair características significativas. Ao invés de calcular diretamente coordenadas em um espaço de alta dimensionalidade, os kernels utilizam o truque do produto interno, permitindo transformar modelos lineares em não lineares. Essa abordagem possibilita que a CNN opere em imagens multicanais, como uma imagem RGB com três canais de cor, ao invés de dados vetoriais, como ocorre em redes tradicionais (ZHAO et al., 2024).

Durante o processo de convolução, o kernel desliza horizontal e verticalmente sobre a imagem de entrada, realizando o produto escalar entre os valores correspondentes do kernel e da imagem. Esse produto é somado, resultando em um único valor escalar que compõe o mapa de características. O procedimento se repete até que o kernel cubra toda a imagem, formando uma

matriz de saída que representa as características extraídas. Por exemplo, na Figura 4 ao aplicar um kernel de 2×2 sobre uma imagem de 4×4, o produto escalar é calculado em cada região coberta, gerando o mapa de características correspondente. Essas operações são essenciais para que a CNN detecte os principais pixels que compõe a imagem (TAYE, 2023).

Figura 4 – Aplicação do kernel nos pixels da imagem e construção do mapa de características



Fonte: Adaptado de Taye (2023).

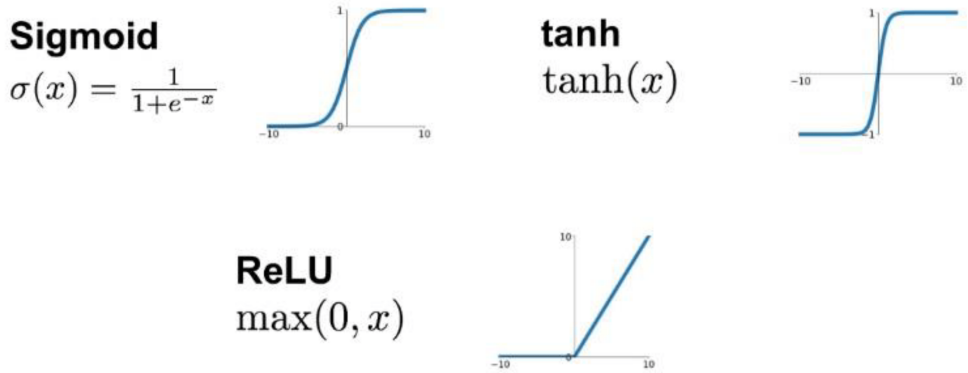
2.1.2 Função de Ativação

Após a camada convolutiva, é introduzida uma camada de não-linearidade, representada pelas funções de ativação. Essas funções são essenciais para introduzir não-linearidade ao mapeamento entre entrada e saída da rede, permitindo que o modelo capture relações mais complexas nos dados. Essa etapa é crucial para a CNN, pois impede que a rede se limite a resolver problemas lineares, aumentando assim sua capacidade de aprendizado e adaptabilidade

(KHANAM et al., 2024).

As funções de ativação mais comuns incluem Sigmoid, Tanh e ReLU, cada uma com características específicas. A Sigmoid mapeia os valores de entrada em uma faixa entre 0 e 1, enquanto a Tanh varia de -1 a 1, ambas úteis em diferentes contextos. No entanto, a ReLU *(Rectified Linear Unit)* é a mais amplamente utilizada em CNNs devido à sua simplicidade computacional e eficiência. Ela converte valores de entrada negativos em zero e mantém os positivos, acelerando o treinamento e reduzindo o consumo de recursos (KHANAM et al., 2024).

Figura 5 – Funções de ativação



Fonte: Taye (2023).

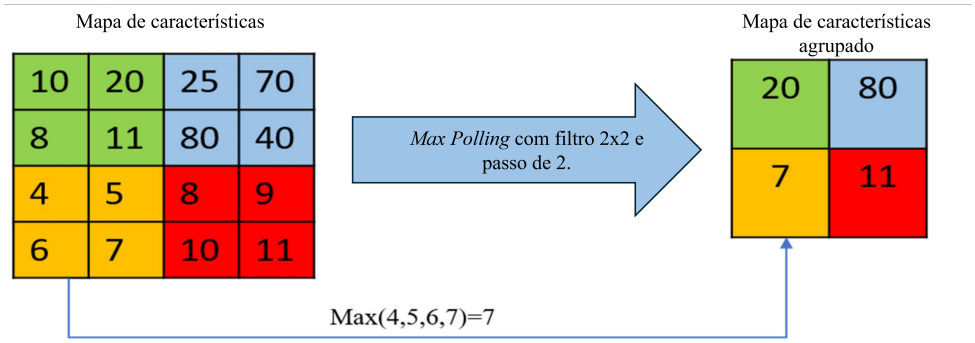
2.1.3 Camada de *Pooling*

A camada de *pooling* desempenha um papel essencial na redução da dimensionalidade dos mapas de características gerados pela camada convolutiva. Essa operação preserva as informações mais relevantes, diminuindo a complexidade computacional das camadas superiores e evitando o risco de *overfitting*. A operação de pooling percorre os dados de entrada utilizando filtros (kernel) que aplicam métodos como *Max Mooling, Min Pooling ou Average Pooling* (média). A função do *pooling* pode ser comparada a uma redução de resolução em processamento de imagens, diminuindo a quantidade de dados, mas retendo informações importantes para as etapas subsequentes (ZAFAR et al., 2022).

Entre as diferentes abordagens, o *Max Pooling* é a mais amplamente utilizada, destacando-se por sua eficiência em extrair características significativas. Nessa técnica, a entrada é dividida em subregiões retangulares, *kernels* de tamanho n x n, e o maior valor de cada bloco é selecionado para o mapa de saída. Isso permite que as características mais expressivas sejam priorizadas, enquanto valores menos importantes são descartados. A Figura 6 mostra a aplicação da operação de *Max Pooling* em um mapa de característica, em que foi utilizado um *kernel* de tamanho 2 x 2

(ZAFAR et al., 2022).

Figura 6 – Aplicação da função Max Pooling para extração de características

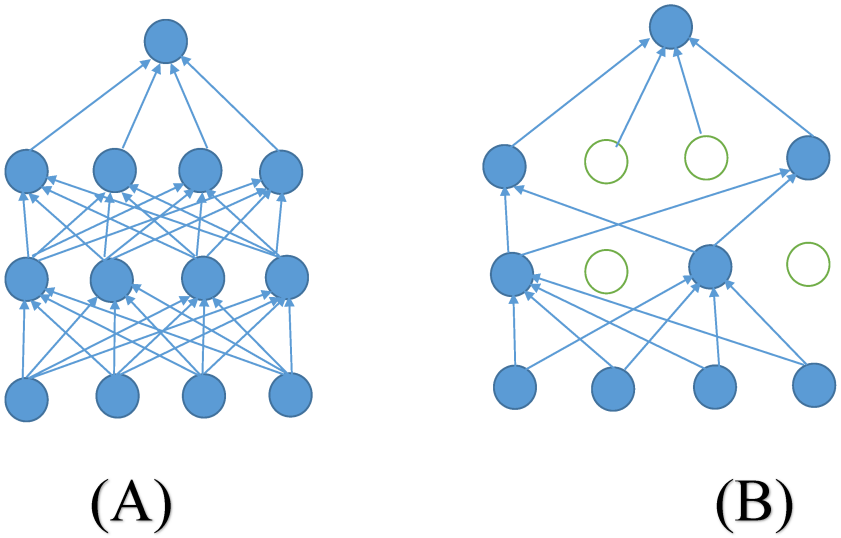


Fonte: Adaptado de Zafar et al. (2022).

2.1.4 *Dropout*

O *Dropout* é uma técnica amplamente utilizada para mitigar o problema de sobreajuste (*overfitting*) em redes neurais convolucionais. Em arquiteturas complexas, com milhares de camadas de processamento, o modelo pode se ajustar excessivamente aos dados de treinamento, comprometendo sua capacidade de generalizar para novos dados. Para solucionar isso, o *Dropout* age desativando aleatoriamente uma porcentagem de neurônios durante o treinamento, com base em uma probabilidade definida, chamada p. Essa abordagem garante que diferentes arquiteturas de rede sejam treinadas para cada amostra, incentivando a rede a não depender de combinações específicas de neurônios e promovendo maior robustez. A Figura 7 (a) apresenta uma rede sem dropout, enquanto que a Figura 7 (b) demonstra uma rede com o neurônio desligado (SALEHIN; KANG, 2023).

Figura 7 – Rede neural padrão (A) e após aplicação de *Dropout* (B)



Fonte: Salehin, Kang (2023).

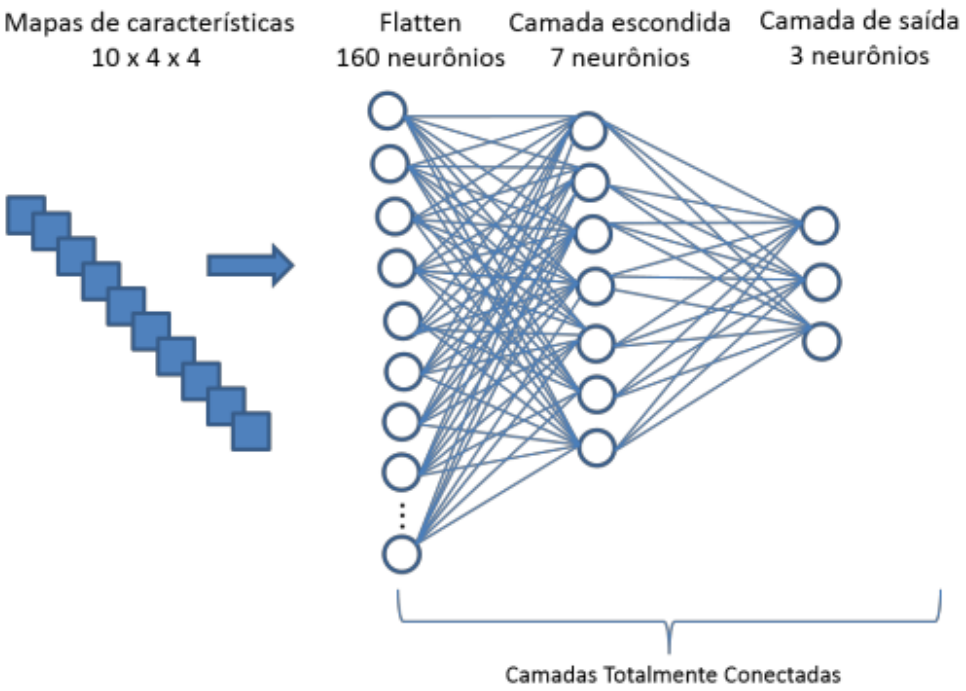
Ao aplicar o *Dropout*, cada iteração do treinamento envolve uma arquitetura ligeiramente diferente, mas todas compartilham os mesmos pesos para os neurônios ativos. Além disso, o *Dropout* melhora o desempenho geral da rede ao introduzir redundância no aprendizado, permitindo que os neurônios "desligados"em uma iteração sejam considerados em outras. Essa estratégia garante que modelos complexos mantenham uma boa capacidade de generalização, mesmo em cenários com grande quantidade de parâmetros (SALEHIN; KANG, 2023).

2.1.5 Camada Totalmente Conectada

A camada totalmente conectada, presente nas Redes Neurais Convolucionais, tem a função de interpretar as características extraídas pelas camadas anteriores. Essa etapa é fundamental para realizar a classificação ou previsão de valores, com base nos dados de treinamento fornecidos. Antes de alcançar a camada totalmente conectada, os mapas de características provenientes das camadas de *pooling*, que apresentam múltiplas dimensões, passam por um processo chamado *flatten*, no qual são redimensionados para uma única dimensão. Esse ajuste permite que as informações sejam conectadas à parte final da rede (SOUSA, 2022).

A Figura 8 mostra uma rede neural recebendo 10 mapas de características com dimensões 4 x 4 cada um. Antes de entrarem na camada totalmente conectada, eles passam pelo processo de *flatten* para que sejam redimensionados. Desse modo, a estrutura da CNN está apta a receber os dados e fazer o processo de treinamento.

Figura 8 – Representação da camada totalmente conectada



Fonte: Adaptado de Souza, (2022).

2.2 Aplicações das Redes Neurais Convolucionais na Engenharia Química

Para demonstrar a aplicação das redes neurais convolucionais, foi selecionado seis artigos de pesquisa e esses divididos em seis grandes áreas da engenharia química. São elas:

1. Processos Físicos-Químicos;
2. Processos de Separação;
3. Modelagem, Controle e Otimização de Processos;
4. Gestão Ambiental e Energias Sustentáveis;
5. Processos Biotecnológicos;
6. Tecnologia de alimentos.

Em cada uma delas será apresentado um artigo em que foi utilizado os conceitos explicados no tópico anterior para resolver problemas e gerar novos conhecimentos.

2.2.1 Processos Físicos-Químicos

O artigo de Davidy (2021) investigou o uso de um queimador de coque de petróleo como fonte de calor para o ciclo termodinâmico de Rankine Orgânico (ORC), com butano como fluido de trabalho. A proposta buscou avaliar se o queimador pode fornecer o fluxo de calor necessário para a operação do ciclo e explorar a eficiência térmica do sistema. Por meio de simulações CFD com o software *Fire Dynamics Simulator* (FDS), o estudo analisou a combustão do coque e validou os resultados contra dados da literatura, mostrando uma temperatura máxima de chama de 1440°C, próxima aos valores reportados.

Além disso, algoritmos de *machine learning*, incluindo ARIMA e CNN, foram usados para prever o comportamento térmico do sistema com base em dados de sensores. A CNN, configurada com oito camadas ocultas e função de ativação ReLU, apresentou excelente desempenho nas previsões, comprovando seu potencial em modelagem de combustão. O ciclo ORC projetado atingiu uma eficiência térmica de 20,4%, e a potência líquida calculada foi de 3472 kW, com um erro relativo de apenas 4,8% em relação à literatura. O estudo destacou ainda a viabilidade de usar o calor gerado para produção de hidrogênio verde, mostrando aplicações potenciais em sistemas sustentáveis e em indústrias que exigem alta eficiência energética (DAVIDY, 2021).

2.2.2 Processos de Separação

O estudo de Li et al. (2019) apresentou um modelo híbrido baseado em CNN e codificadores automáticos profundos (DAE) para diagnóstico de falhas em processos de destilação. A proposta visa superar os desafios apresentados pela alta dimensionalidade dos dados e pelo forte acoplamento de variáveis nesses sistemas. O modelo combina a capacidade das CNNs para redução de dimensionalidade e extração de características com a habilidade dos DAEs para classificação, formando um sistema robusto e eficiente. O trabalho utilizou como estudo de caso o processo de despropanização, onde componentes de gás liquefeito de petróleo (GLP) são separados.

O modelo foi treinado e validado com dados simulados, envolvendo 40 variáveis e cinco condições de operação, incluindo estados normais e falhas como mudanças na composição da alimentação e interrupções no sistema de aquecimento. A arquitetura final do modelo incluiu três camadas convolucionais, quatro camadas de ativação, duas camadas de *pooling*, uma camada de *dropout* e camadas totalmente conectadas. O modelo híbrido alcançou uma precisão de 92,2% no diagnóstico de falhas, superando métodos individuais baseados apenas em CNNs ou DAEs.

Além disso, o uso de funções de ativação PReLU e pooling máximo otimizou o desempenho (LI et al., 2019).

Os resultados destacam o potencial do modelo CNN–DAE para monitoramento em tempo real e diagnóstico de falhas em colunas de destilação, contribuindo para maior segurança e eficiência operacional. No entanto, os autores ressaltam que a aplicação em dados industriais reais ainda demanda estudos adicionais devido a problemas como ruído nos sinais e disponibilidade limitada de dados anômalos (LI et al., 2019).

2.2.3 Modelagem, Controle e Otimização de Processos

O artigo de Theisen et al. (2023) apresentou um *framework* CNN (abordagem estruturada que utilizou as redes Faster R-CNN e ResNet-50, além de um conjunto de dados formados por 47 tipos diferentes de operações unitárias) para a digitalização de diagramas de fluxo de processos (PFDs). O objetivo foi superar os desafios impostos pela variabilidade de símbolos e estilos presentes nesses diagramas, tornando-os acessíveis para ferramentas digitais e permitindo sua integração em sistemas de inteligência artificial. O modelo combina detecção de objetos via CNN com um algoritmo de busca de conectividade para criar representações gráficas dos fluxogramas.

Foram utilizados mais de 1000 PFDs provenientes de fontes diversificadas, como revistas científicas e livros. O framework inclui três etapas principais: detecção de objetos (operações unitárias, textos e setas) por meio de uma CNN do tipo Faster R-CNN, identificação de conectividades entre as operações por um algoritmo pixel-based, e conversão das informações em um grafo representando o diagrama. O uso de aprendizado por transferência a partir do dataset COCO (um conjunto de dados que contém mais de 200.000 imagens para diversas tarefas de detecção de objetos) permitiu que o modelo lidasse com os desafios impostos pela pequena dimensão do conjunto de dados. O trabalho demonstrou o potencial das CNNs para digitalizar diagramas técnicos de forma eficaz, reduzindo custos e melhorando a interoperabilidade no setor químico (THEISEN et al., 2023)

2.2.4 Gestão Ambiental e Energias Sustentáveis

A pesquisa de O'Donovan, Giannetti e Pleydell-Pearce (2024) explorou o potencial da visão computacional, com destaque para CNN, na promoção da sustentabilidade na produção de aço, uma indústria responsável por 7% das emissões globais de CO_2. O objetivo foi identificar

aplicações de CNNs que possam reduzir impactos ambientais, melhorar a eficiência dos processos e aumentar a segurança no ambiente de trabalho. O estudo realizou uma revisão literária das principais aplicações de visão computacional na indústria siderúrgica e propôs um roteiro baseado nos princípios da Indústria 4.0 para integrar essas tecnologias de forma sustentável.

Entre as aplicações discutidas, a detecção de falhas em superfícies de aço, o monitoramento de temperatura em altos-fornos e a automação de processos de remoção de escória são destacadas como soluções eficazes. Por exemplo, a CNN combinada com algoritmos de aprendizado profundo foi usada para prever trajetórias de remoção de escória com precisão superior a 90%. Outra aplicação foi o uso de sistemas de visão para identificar defeitos em superfícies de aço com taxas de acurácia acima de 95%, reduzindo desperdícios e aumentando a qualidade do produto. Os resultados mostram que as CNNs não apenas aprimoram o desempenho operacional, mas também reduzem o consumo de energia e os riscos associados à exposição humana a equipamentos perigosos (O'DONOVAN; GIANNETTI; PLEYDELL-PEARCE, 2024).

2.2.5 Processos Biotecnológicos

Zhong et al. (2021) exploraram o uso de Redes Neurais Convolucionais baseadas em imagens moleculares para desenvolver modelos de Relação Quantitativa Estrutura-Atividade (QSARs) capazes de prever a reatividade de contaminantes com radicais hidroxila (OH). A pesquisa utilizou dados de 1.089 compostos orgânicos e aplicou aprendizado por transferência e aumento de dados para superar a limitação de dados e melhorar a robustez dos modelos. A abordagem utilizou a arquitetura DenseNet121 (arquitetura de rede neural convolucional composta por 121 camadas que foi utilizado para lidar com o conjunto de dados) pré-treinada e incorporou técnicas como rotação e espelhamento de imagens moleculares.

Os resultados demonstraram que os modelos baseados em CNN superaram os métodos tradicionais baseados em descritores moleculares e impressões digitais moleculares, com redução do RMSE nos dados de teste para um intervalo de 0,284–0,339.

2.2.6 Tecnologia de alimentos

Khaparde et al. (2023) propuseram um modelo baseado em redes neurais convolucionais para detectar doenças em folhas de batata, com foco em doenças como a pinta-preta *(early blight)* e a requeima *(late blight)*. Usando o dataset "Plant Village"do Kaggle, composto por mais de 2000 imagens de folhas saudáveis e infectadas, o estudo treinou um modelo de CNN

para identificar e classificar as folhas com alta precisão. A metodologia incluiu processamento de imagens para extração de características como cor, textura e forma, seguido de segmentação e análise por CNNs. O modelo foi estruturado com camadas convolucionais, pooling, ReLU para ativação e camadas totalmente conectadas para classificação. Durante os testes, o sistema atingiu 91,41% de precisão ao diferenciar folhas saudáveis das doentes.

As CNNs foram essenciais para identificar padrões visuais específicos associados às doenças, tornando o diagnóstico rápido e acessível. Os autores destacaram a relevância do modelo para aumentar a produtividade agrícola, oferecendo aos agricultores uma ferramenta prática para monitorar doenças e implementar medidas corretivas com agilidade (KHAPARDE et al., 2023).

3 ESTUDO DE CASO COMPUTACIONAL: IMPLEMENTAÇÃO EM PYTHON

3.1 Definição do Problema

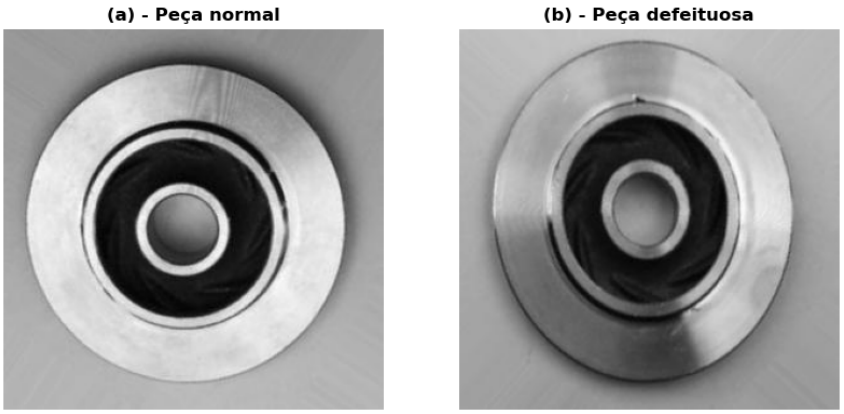
O processo de fundição, amplamente utilizado em indústrias automotivas, aeroespaciais e de engenharia, consiste na introdução de metal líquido em moldes que contêm cavidades no formato desejado. Após solidificado, o metal assume a geometria do molde, resultando em peças com propriedades específicas. Apesar de sua versatilidade, esse processo é propenso a defeitos, como porosidade, inclusão de areia, rachaduras e deformações. Esses defeitos impactam diretamente a qualidade final dos produtos e a eficiência das operações industriais, tornando a inspeção de qualidade uma etapa indispensável no controle do processo (SAI; VINOD; SOWMYA, 2017).

A escolha da fundição como foco deste estudo justifica-se por sua relevância no setor industrial e pela complexidade associada à inspeção manual de qualidade, que é demorada e sujeita a erros humanos. A aplicação de Visão Computacional surge como uma solução promissora para identificar automaticamente defeitos em peças fundidas, reduzindo custos operacionais e melhorando a precisão da análise (LIN et al., 2018).

A base de dados já foi disponibilizada separada em dois conjuntos: treino e teste, com 6633 imagens para treinamento e 715 para teste. Ambas as pastas possuem subdivisões que indicam as duas classes presentes: peças normais e defeituosas. Essa separação prévia foi realizada pelo próprio autor do repositório no Kaggle, o que facilitou a organização e o carregamento automático dos dados. Considerando o total de imagens, aproximadamente 90% foram destinadas ao treinamento e 10% ao teste. No total, 42,69% das imagens representam peças normais e 57,31% peças com defeito.

Neste estudo, será utilizada uma base de dados real extraída da plataforma Kaggle, composta por imagens de peças fundidas com e sem defeitos que está apresentada na Figura 9. Ela foi disponibilizada por Ravirajsinh Dabhi (DABHI, 2020) e trata-se de um problema de classificação binária que foi resolvido na ferramenta Google Colaboratory (Google, 2025). A metodologia envolve o treinamento de modelos de Rede Neural Convolucional para classificar as peças e identificar falhas estruturais. Essa aplicação prática exemplifica como a inteligência artificial pode transformar o controle de qualidade em processos químicos, promovendo maior eficiência e sustentabilidade na produção industrial.

Figura 9 – Peças de fundição utilizadas para o treinamento da rede neural convolucional



Fonte: Autoria Própria (2025).

3.2 Bibliotecas para manipulação das imagens

As bibliotecas utilizadas neste estudo de caso foram selecionadas para cobrir todas as etapas do desenvolvimento da CNN, desde o carregamento das imagens até a avaliação dos resultados. O *TensorFlow Keras* (TensorFlow Developers, 2024) foi utilizado para a construção, treinamento e validação do modelo, permitindo a definição das camadas convolucionais, de *pooling* e densas por meio do módulo *Keras.layers*. O carregamento e a normalização das imagens foram realizados pelo *Keras.preprocessing.image*, enquanto o *Keras.utils* possibilitou a visualização da arquitetura do modelo. Além disso, *Keras.backend* foi empregado para limpar a sessão entre os treinamentos, evitando o consumo excessivo de memória.

A avaliação do modelo foi realizada com auxílio da *Matplotlib* (HUNTER, 2007) e *Seaborn* (WASKOM, 2021), permitindo a construção de gráficos da evolução da perda e da acurácia ao longo das épocas. A biblioteca *JSON* (Python Software Foundation, 2024b) foi utilizada para armazenar os resultados do treinamento em arquivos estruturados, garantindo a organização e reprodutibilidade dos experimentos.

Além das bibliotecas voltadas para aprendizado de máquina e análise de dados, outras ferramentas auxiliaram na organização do ambiente de execução. O módulo *zipfile* (Python Software Foundation, 2024d) foi empregado para extrair o conjunto de dados compactado, possibilitando seu uso no treinamento do modelo. A biblioteca *os* (Python Software Foundation, 2024a) foi essencial para a navegação entre diretórios e a manipulação de caminhos de arquivos de forma dinâmica. Para evitar mensagens desnecessárias durante a execução, a biblioteca *Warnings* (Python Software Foundation, 2024c) foi utilizada para suprimir avisos. Por fim, os resultados do treinamento foram armazenados e manipulados em tabelas estruturadas com a

biblioteca *Pandas* (TEAM, 2024), facilitando a exportação e análise das métricas do modelo.

3.3 Estrutura da Rede Neural Convolucional

A base de dados utilizada no treinamento da rede neural já estava separada em dois conjuntos: treino e teste. As imagens foram carregadas utilizando a função `flow_from_directory`, que automaticamente identifica as classes a partir da estrutura de diretórios.

Para garantir que o modelo aprenda os padrões das imagens de maneira eficiente, foi aplicada uma normalização, convertendo os valores dos pixels para a escala entre 0 e 1, além disso todas elas foram convertidas para escala de cinza reduzindo a complexidade computacional do modelo. Essa abordagem melhora a convergência da rede ao eliminar a necessidade de processar múltiplos canais de cores, focando exclusivamente nas variações de intensidade dos defeitos identificados nas peças de fundição.

O estudo envolve o treinamento de duas arquiteturas de redes, diferenciadas pela quantidade de camadas densas. A primeira arquitetura, Tabela 01, conta com uma única camada densa de 128 neurônios. Já a segunda arquitetura, Tabela 02, apresenta duas camadas densas de 128 neurônios cada. Ambas as redes utilizam camadas convolucionais com filtros 3×3 e função de ativação ReLU, seguidas por camadas de MaxPooling 2×2, que reduzem a dimensionalidade dos dados. Para mitigar o *overfitting*, foi aplicada a técnica de *Dropout* com uma taxa de 20%, desativando aleatoriamente uma fração dos neurônios durante o treinamento, reduzindo a dependência excessiva da rede em determinados padrões.

Tabela 1 – Estrutura da CNN e número de parâmetros treináveis com uma camada densa.

Camada	**Tipo**	**Saída**	**Parâmetros treináveis**
1	conv2d (3×3)	300 × 300 × 32	320
2	`max_pooling2d` (2×2)	150 × 150 × 32	0
3	`conv2d_1` (3×3)	150 × 150 × 32	9.248
4	`max_pooling2d_1` (2×2)	75 × 75 × 32	0
5	flatten	180.000	0
6	dense	224	23.040.128
7	dropout (20%)	224	0
8	`dense_1` (Sigmoide)	1	129
Total			**23.049.825**

Fonte: Autoria Própria (2025).

Tabela 2 – Estrutura da CNN e número de parâmetros treináveis com duas camadas densas.

Camada	**Tipo**	**Saída**	**Parâmetros treináveis**
1	conv2d (3×3)	300 × 300 × 32	320
2	`max_pooling2d` (2×2)	150 × 150 × 32	0
3	`conv2d_1` (3×3)	150 × 150 × 32	9.248
4	`max_pooling2d_1` (2×2)	75 × 75 × 32	0
5	flatten	180.000	0
6	dense	224	23.040.128
7	dropout (20%)	224	0
8	`dense_1`	224	16.512
9	`dropout_1` (20%)	224	0
10	`dense_2` (Sigmoide)	1	129
Total			**23.066.337**

Fonte: Autoria Própria (2025).

3.3.1 Escolha dos parâmetros da rede

A definição dos parâmetros de uma rede neural influencia diretamente o seu desempenho, afetando tanto a capacidade de aprendizado quanto a generalização do modelo. Nesta subseção, serão justificadas as escolhas relacionadas ao número de filtros, tamanho do kernel, funções de ativação, número de neurônios, taxa de *Dropout*, função de perda, otimizador e métrica de avaliação, destacando sua importância no contexto do problema abordado.

A rede neural utilizada no estudo contém 32 filtros nas camadas convolucionais. A escolha de 32 filtros é baseada em práticas recomendadas para redes profundas, onde um número menor de filtros nas camadas iniciais permite capturar padrões básicos, como bordas e texturas, reduzindo o custo computacional. À medida que a profundidade da rede aumenta, um número maior de filtros pode ser utilizado para extrair características mais complexas (GOODFELLOW; BENGIO; COURVILLE, 2016). Em relação ao tamanho do filtro adotado foi de 3×3. Esse tamanho permite capturar características locais mantendo um equilíbrio entre complexidade e eficiência (SIMONYAN, 2015).

Para as funções de ativação, utilizou-se ReLU (*Rectified Linear Unit*) nas camadas convolucionais e densas. Ela é definida como:

$$f(x) = \max(0, x) \tag{3.1}$$

Essa ativação acelera o aprendizado e permite a modelagem de relações não lineares (NAIR, 2010). Já na camada de saída, utilizou-se a função sigmoide, pois o problema trata de

classificação binária. A função sigmoide é definida como:

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.2}$$

Ela mapeia a saída para o intervalo (0,1), permitindo que a rede modele probabilidades (BISHOP; NASRABADI, 2006).

Em relação as camadas densas elas possuem 128 neurônios. Essa escolha segue um padrão utilizado em arquiteturas de *deep learning* em que são escolhidos potências de dois (32, 64, 128, 256...) porque garantem boa alocação de memória e paralelismo computacional (GOODFELLOW; BENGIO; COURVILLE, 2016). A definição do número de neurônios foi feita empiricamente, buscando capturar representações abstratas sem aumentar excessivamente a complexidade do modelo. Para reduzir o risco de *overfitting*, utilizou-se a técnica de *dropout*, com taxa de 20% (0,2), o que significa que 20% dos neurônios são desativados aleatoriamente durante o treinamento. Essa abordagem impede que a rede memorize padrões específicos dos dados de treino, melhorando sua capacidade de generalização (SRIVASTAVA et al., 2014).

A função de perda utilizada foi a *binary crossentropy*, recomendada para problemas de classificação binária. Essa função maximiza a separação entre as duas classes e melhora a estabilidade do aprendizado (MURPHY, 2012). O otimizador escolhido foi o Adam *(Adaptive Moment Estimation)*. Segundo Kingma e Ba (2014), esse método é computacionalmente eficiente, requer pouca memória, é invariante ao reescalonamento diagonal dos gradientes e adequado para problemas de grande escala em termos de dados e parâmetros. Devido a essas características, Adam é amplamente utilizado em redes neurais profundas, oferecendo um bom equilíbrio entre velocidade de convergência e estabilidade do treinamento (KERAS, 2024b).

Por fim, a métrica de avaliação utilizada foi a acurácia, que mede a proporção de previsões corretas em relação ao total de amostras avaliadas. Essa métrica é amplamente utilizada em problemas de classificação para avaliar o desempenho do modelo, sendo que o contexto de redes neurais, a acurácia fornece um indicador direto da eficácia do modelo, auxiliando na comparação entre diferentes abordagens de treinamento (KERAS, 2024a).

Embora a escolha dos parâmetros da rede neural tenha seguido referências da literatura e práticas consolidadas, é importante destacar que métodos automatizados de ajuste de hiper-parâmetros são amplamente utilizados para otimizar os modelos. Técnicas como Otimização Bayesiana, Grid Search, Randomized Search e Optuna permitem explorar sistematicamente diferentes configurações de parâmetros, visando maximizar o desempenho da rede. No entanto, esses métodos exigem alto poder computacional, pois envolvem múltiplos treinamentos do

modelo com diferentes combinações de hiperparâmetros (Amazon Web Services, 2024). Dada a limitação de recursos computacionais disponíveis para este estudo, a escolha dos parâmetros foi realizada com base em referências da literatura e experimentação empírica.

3.3.2 Quantidade de pesos treináveis

A quantidade total de pesos treináveis de uma rede neural convolucional depende da estrutura das camadas convolucionais e densas, levando em consideração o número de filtros, o tamanho do *kernel*, e as conexões entre camadas. Para a primeira arquitetura, que contém apenas uma camada densa, o número total de pesos treináveis é de 23.049.825, enquanto para a segunda arquitetura, que inclui duas camadas densas, esse número sobe para 23.066.337.

Os cálculos começam pela primeira camada convolucional *conv2d*, que possui 32 filtros, um *kernel* de 3×3 e opera sobre imagens de entrada com um canal em escala de cinza. O número de parâmetros dessa camada é obtido pela equação:

$$\text{Parâmetros} = (\text{Tamanho do Kernel} \times \text{Canais de Entrada} + 1) \times \text{Filtros} \tag{3.3}$$

O termo + 1 na equação refere-se a um *bias* adicionado automaticamente pela rede neural a cada filtro, permitindo que o modelo aprenda um deslocamento independente para cada saída convolucional (ZHANG et al., 2020). Aplicando os valores para a camada ‘conv2d‘:

$$(3 \times 3 \times 1 + 1) \times 32 = (9 + 1) \times 32 = 320 \tag{3.4}$$

A segunda camada convolucional `conv2d_1` segue a mesma lógica, mas recebe 32 canais de entrada (saída da primeira camada), resultando em:

$$(3 \times 3 \times 32 + 1) \times 32 = (288 + 1) \times 32 = 9.248 \tag{3.5}$$

Após as camadas convolucionais e de *pooling*, os dados são achatados na camada *Flatten*, gerando um vetor de 180.000 elementos que se conecta à primeira camada densa *(dense)*. Essa camada possui 224 neurônios, e cada um recebe 180.000 conexões, totalizando:

$$(180.000 + 1) \times 128 = 23.040.128 \tag{3.6}$$

Na primeira arquitetura, essa camada densa é seguida diretamente pela camada de saída `dense_1`, que possui apenas um neurônio e recebe 224 conexões mais um bias, resultando em:

$$(128+1)\times 1 = 129 \tag{3.7}$$

Somando todos os parâmetros, temos:

$$320 + 9.248 + 23.040.128 + 129 = 23.049.825 \tag{3.8}$$

Na segunda arquitetura, foi adicionada uma segunda camada densa intermediária com 224 neurônios, aumentando a complexidade do modelo. Como ela recebe 224 conexões da camada anterior e um bias para cada neurônio, o cálculo dos novos parâmetros adicionados é:

$$(128+1)\times 128 = 16.512 \tag{3.9}$$

Esse acréscimo faz com que o número total de parâmetros treináveis da segunda arquitetura seja:

$$23.049.825 + 16.512 = 23.066.337 \tag{3.10}$$

Assim, observa-se que a inclusão de uma segunda camada densa aumenta significativamente o número de parâmetros ajustáveis, o que pode impactar tanto a capacidade de aprendizado do modelo quanto seu tempo de treinamento.

3.4 Testes aplicados

A fim de avaliar o desempenho da rede neural convolucional, foram realizados quatro treinamentos distintos, variando a quantidade de camadas densas ocultas e o número de épocas. O objetivo dessas variações é compreender o impacto da profundidade da rede e da quantidade de atualizações dos pesos na capacidade do modelo de aprender padrões relevantes nos dados. A Tabela 03 apresenta a configuração de cada caso analisado, onde os testes foram estruturados considerando redes com uma ou duas camadas densas ocultas e 5 ou 10 épocas de treinamento.

Tabela 3 – Cenários experimentais para avaliação da aprendizagem do modelo

Caso	**Número de Camadas Densas**	**Épocas**
1	1	5
2	1	10
3	2	5
4	2	10

Fonte: Autoria Própria (2025).

Para garantir uma avaliação mais confiável do desempenho dos modelos, cada cenário experimental foi executado 10 vezes. Essa repetição tem o objetivo de reduzir a influência de variações aleatórias inerentes ao treinamento de redes neurais, como inicialização dos pesos e ordem dos lotes de dados. Dessa forma, os resultados obtidos refletem melhor a tendência real de desempenho dos modelos, minimizando o impacto de flutuações estatísticas.

Os resultados dos treinamentos foram avaliados com base em métricas estatísticas, incluindo média, desvio padrão da acurácia em cada configuração.

3.5 Resultados Obtidos

O treinamento de uma rede neural ocorre por meio de sucessivas atualizações dos pesos dos neurônios, a fim de minimizar o erro da previsão. Cada atualização completa, onde todos os dados do conjunto de treinamento passam pela rede, é chamada de época. No presente estudo, o treinamento foi realizado com um tamanho de lote *(batch size)* de 32, ou seja, as imagens foram processadas em lotes de 32 amostras por vez antes da atualização dos pesos. O *batch size* controla como os dados são divididos durante o treinamento. Como o conjunto é composto por 6633 imagens, e um *batch size* de 32, cada época de treinamento foi composta por 208 lotes.

Isso significa que, em cada época, os dados foram alimentados na rede em 208 pacotes consecutivos de 32 imagens, com os pesos sendo atualizados ao final de cada lote. Esse método reduz a carga computacional, otimizando o uso da memória e tornando o treinamento mais eficiente.

3.5.1 Análise da acurácia

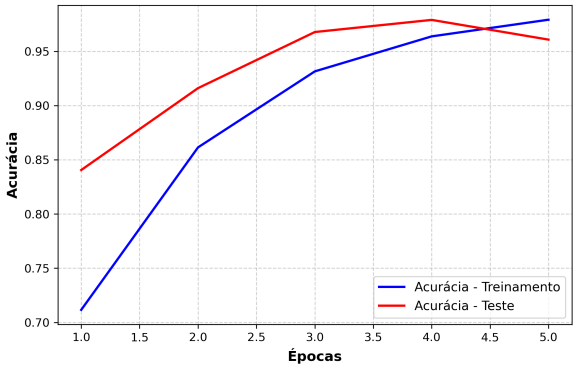
A Tabela 4 apresenta os resultados detalhados dos treinamentos realizados para os quatro casos analisados. São fornecidos os valores acurácia no conjunto de teste.

Tabela 4 – Resultados obtidos nos treinamentos com os dados de teste

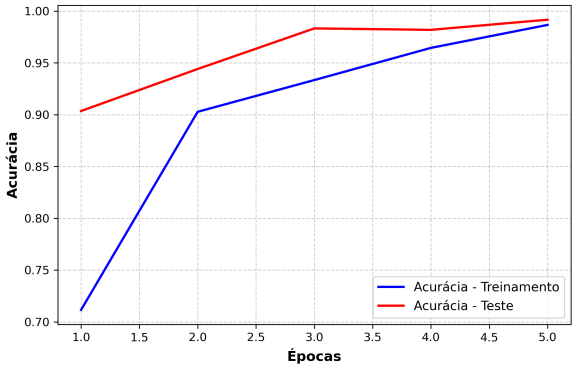
Teste	**Caso 1**	**Caso 2**	**Caso 3**	**Caso 4**
1	0,9958	0,9930	0,9790	0,9986
2	0,9832	0,9497	0,9958	0,9902
3	0,9874	0,9944	0,9678	0,9958
4	0,9804	0,9930	0,9916	0,9860
5	0,9916	0,9944	0,9832	0,9986
6	0,9846	0,9832	0,9930	1,0000
7	0,9385	0,9916	0,9958	0,9916
8	0,9902	0,9972	0,9944	0,9972
9	0,9692	0,9888	0,9958	0,9958
10	0,9608	0,9916	0,9916	0,9958

Fonte: Autoria Própria (2025).

As Figuras 10a, 10b, 11a e 11b apresentam os gráficos da evolução da acurácia para os quatro modelos desenvolvidos. Esses gráficos permitem avaliar o comportamento dos algoritmos em relação ao aprendizado ao longo das épocas e sua capacidade de generalização nos dados de teste.



(a) Evolução da acurácia - Modelo 1

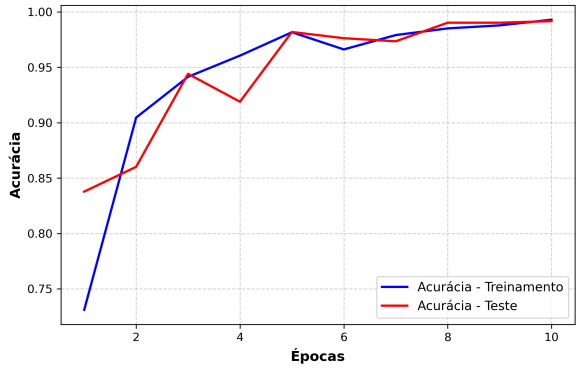


(b) Evolução da acurácia - Modelo 3

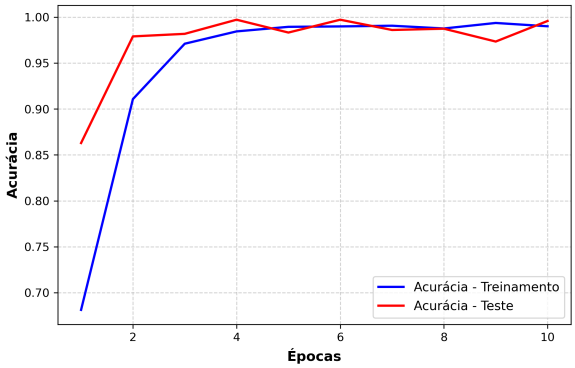
Figura 10 – Acurácia nos modelos com cinco épocas de treinamento

Fonte: Autoria Própria (2025).

Observando os gráficos dos modelos treinados com cinco épocas, nota-se que tanto as curvas de treino quanto de teste não atingiram um comportamento estável. A acurácia ainda apresentava crescimento ao final da última época, sugerindo que o modelo não convergiu totalmente. Esse comportamento indica que seria recomendável o uso de um número maior de épocas de treinamento, permitindo que a acurácia evoluísse até atingir um patamar de estabilidade mais adequado.



(a) Evolução da acurácia - Modelo 2



(b) Evolução da acurácia - Modelo 4

Figura 11 – Acurácia nos modelos com dez épocas de treinamento

Fonte: Autoria Própria (2025).

Já nos modelos que utilizaram dez épocas, verifica-se que, a partir da oitava iteração, tanto a acurácia de treino quanto a de teste apresentaram estabilidade. Esse comportamento indica que os algoritmos alcançaram um ponto de convergência, ou seja, o treinamento foi suficiente para que o desempenho não sofresse mais variações relevantes, evidenciando que a rede neural atingiu um patamar de aprendizado consistente.

Com o objetivo de determinar qual modelo apresentou o melhor desempenho de forma estatisticamente significativa, foi realizado uma análise de variância (ANOVA). Esse teste é apropriado para este estudo, pois permite comparar as médias das acurácias dos quatro casos analisados, levando em conta a variabilidade entre os treinamentos. A ANOVA é indicada quando se deseja avaliar se há diferenças significativas entre três ou mais grupos. Dessa forma, a aplicação desse teste estatístico fornecerá uma base rigorosa para a interpretação dos resultados obtidos (HU et al., 2025).

3.5.2 Precisão, Recall e F1-Score

Além da acurácia, foram calculadas métricas complementares, como precisão, recall e F1-Score que estão apresentadas na Tabela 5.

Tabela 5 – Resultados de desempenho para os Casos 1, 2, 3 e 4 no Teste 10.

Caso	**Precisão**	**Recall**	**F1-Score**
Caso 1	0,9520	0,9683	0,9586
Caso 2	0,9902	0,9918	0,9910
Caso 3	0,9888	0,9934	0,9910
Caso 4	0,9951	0,9959	0,9955

Os resultados mostraram que, em todos os casos avaliados, os valores dessas métricas

mantiveram-se elevados, reforçando a robustez das arquiteturas propostas. Tais indicadores são relevantes, pois cada um deles enfatiza diferentes aspectos da performance do classificador, como a capacidade de identificar corretamente os defeitos (recall) ou de manter a consistência das classificações positivas (precisão). Dessa forma, a análise conjunta dessas métricas complementa a avaliação pela acurácia, permitindo uma interpretação mais abrangente dos resultados (SOKOLOVA; LAPALME, 2009).

3.5.3 Aplicação de testes estatísticos para avaliação do treinamento

Para avaliar a significância estatística das diferenças entre os modelos treinados, foi realizada uma Análise de Variância (ANOVA). Esse teste é apropriado para comparar médias de três ou mais grupos independentes e verificar se ao menos um deles difere significativamente dos demais. A hipótese nula (H_0) assume que não há diferença significativa entre os modelos, enquanto a hipótese alternativa (H_a) sugere que ao menos um modelo possui desempenho estatisticamente distinto dos demais (MONTGOMERY, 2017).

Inicialmente, foi aplicado o teste de Shapiro-Wilk para verificar a normalidade dos dados, obtendo um p-valor de 0.0584. Como este valor é superior ao nível de significância de 0.05, não há evidências para rejeitar a normalidade dos dados. Em seguida, foi realizado o teste de Levene para avaliar a homogeneidade das variâncias, resultando em um p-valor de 0.5085, indicando que as variâncias entre os modelos são homogêneas. Essas condições justificam a aplicação da ANOVA (FIELD, 2024).

A ANOVA indicou um valor de p-valor de 0.0339, o que permite rejeitar ao nível de significância de 5%. Isso sugere que há diferenças estatisticamente significativas entre pelo menos dois dos modelos analisados. Para identificar quais modelos apresentam diferenças significativas entre si, foi aplicado o teste de Tukey, um método de comparações múltiplas apropriado para esse tipo de análise (PIMENTEL-GOMES, 2023).

Os resultados do teste de Tukey estão apresentados na Tabela 6. Esse teste permite verificar quais comparações entre pares de modelos apresentam diferenças estatisticamente significativas.

Os valores obtidos indicam que há uma diferença estatisticamente significativa entre os Modelos 1 e 4, uma vez que o p-valor da comparação entre esses dois grupos foi 0,0218, inferior ao nível de significância adotado ($\alpha = 0,05$). Esse resultado sugere que o desempenho do Modelo 4 difere significativamente do Modelo 1.

Tabela 6 – Comparação múltipla de médias pelo teste de Tukey ($\alpha = 0,05$).

Grupo 1	**Grupo 2**	**Diferença de Médias**	**p-valor**	**Diferença Significativa?**
Modelo 1	Modelo 2	0,0166	0,1831	Não
Modelo 1	Modelo 3	0,0145	0,2915	Não
Modelo 1	**Modelo 4**	**0,0245**	**0,0218**	**Sim**
Modelo 2	Modelo 3	-0,0022	0,9929	Não
Modelo 2	Modelo 4	0,0078	0,7655	Não
Modelo 3	Modelo 4	0,0100	0,6035	Não

Por outro lado, as comparações entre os Modelos 2, 3 e 4 não apresentaram diferenças estatisticamente significativas, uma vez que os p-valores das respectivas comparações foram todos superiores a 0,05. Portanto, pode-se concluir que os Modelos 2, 3 e 4 não diferem significativamente entre si em termos de desempenho.

3.5.4 Avaliação da matriz de confusão

Embora a Análise de Variância (ANOVA) tenha indicado que o Modelo 4 apresentou um desempenho significativamente superior ao Modelo 1, não foram observadas diferenças estatisticamente relevantes entre os Modelos 2, 3 e 4. Dessa forma, qualquer um deles poderia ser escolhido. Para auxiliar nessa decisão, a matriz de confusão torna-se uma ferramenta essencial, pois permite visualizar a relação entre as previsões do modelo e os valores reais, identificando erros e acertos. A utilização dessa ferramenta na avaliação de modelos de classificação é amplamente reconhecida por sua eficácia em quantificar o desempenho preditivo e a confiabilidade dos classificadores (FERNANDEZ et al., 2018).

A matriz de confusão é uma ferramenta que possibilita a análise do desempenho de um sistema de classificação ao comparar as previsões do modelo com as classes reais. Os valores que compõem essa matriz são obtidos ao submeter o conjunto de teste ao modelo e analisar os resultados da predição. Dessa forma, os valores da matriz são classificados em quatro categorias: Verdadeiro Positivo (VP), que representa instâncias corretamente classificadas como pertencentes à classe positiva; Falso Positivo (FP), que indica elementos incorretamente classificados como positivos; Falso Negativo (FN), que representa casos positivos classificados erroneamente como negativos; e Verdadeiro Negativo (VN), que corresponde a instâncias corretamente identificadas como negativas (FERNANDEZ et al., 2018). O posicionamento desses resultados na matriz está representado na Figura 12.

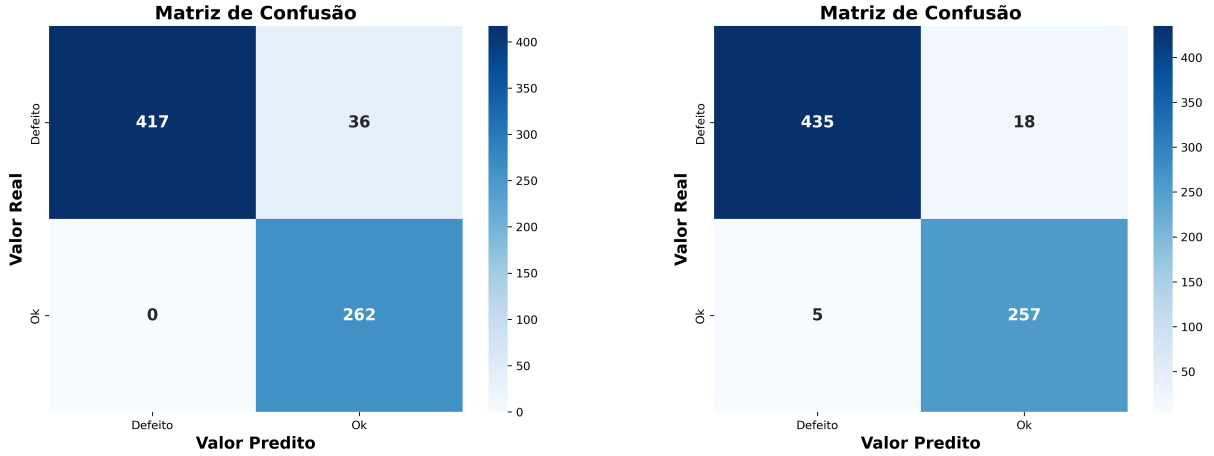
Figura 12 – Estrutura para avaliação da matriz de confusão

Fonte: (GUIMARÃES, 2024).

Dentre essas categorias, os Falsos Positivos (FP) são especialmente críticos em aplicações industriais, pois indicam itens defeituosos que foram classificados como bons. Esse tipo de erro pode comprometer o controle de qualidade e gerar impactos significativos na linha de produção. Assim, para a análise dos Modelos 2, 3 e 4 foi considerada a quantidade de Falsos Positivos gerados, de modo a selecionar o modelo que apresente menor incidência desse erro.

Para cada modelo, foram executados dez testes, e para cada um deles plotada uma matriz de confusão. O critério de escolha do modelo mais eficiente baseou-se na minimização do número de falsos positivos, garantindo um maior controle sobre a confiabilidade das previsões realizadas.

As Figura 13a e 13b a seguir apresentam os resultados obtidos para o Caso 2 teste 2 e Caso 3 teste 3. Eles foram plotados porque ambos os modelos obtiveram os maiores valores de Falsos Positivos, 36 e 18, respectivamente. No contexto desse estudo de caso, esse tipo de erro pode levar a decisões equivocadas, resultando na aprovação de peças defeituosas que, na realidade, não atendem aos padrões de qualidade. Por isso, não é recomendado a escolha dos Casos 2 e 3 para uma implementação industrial.



(a) Segundo teste do Caso dois

(b) Terceiro teste do Caso três

Figura 13 – Quantidade de Falsos Positivos dos Casos dois e três

A Tabela 7 apresenta a quantidade de Falsos Positivos encontrados ao longo dos dez testes realizados para os Casos 2, 3 e 4. Nota-se que, com exceção dos testes 2 e 7, o modelo do Caso 4 apresentou um desempenho superior, pois obteve um menor número de Falsos Positivos em comparação com os demais modelos. Esse resultado indica uma maior precisão do modelo na classificação correta das amostras, reduzindo o risco de decisões equivocadas baseadas em previsões incorretas.

Tabela 7 – Número de Falsos Positivos obtidos nos Casos 2, 3 e 4

Teste	**Caso 2**	**Caso 3**	**Caso 4**
1	4	14	0
2	2	1	6
3	36	18	0
4	3	3	2
5	4	12	0
6	12	3	0
7	2	2	4
8	1	4	0
9	8	2	1
10	4	6	2

Fonte: Autoria Própria (2025).

Assim, com base na análise estatística e na avaliação da matriz de confusão, o modelo selecionado para implementação foi o do Caso 4. Essa escolha foi fundamentada no fato de que, na maioria dos testes, esse modelo apresentou a menor quantidade de Falsos Positivos, conforme evidenciado na Tabela 7. Embora nos testes 2 e 7 seu desempenho tenha sido inferior, o resultado geral demonstra uma tendência de maior precisão e confiabilidade na predição. A adoção do

modelo do Caso 4 minimiza o impacto de erros críticos no processo industrial, garantindo uma melhor tomada de decisão e aumentando a eficiência operacional. Dessa forma, a matriz de confusão se mostra uma ferramenta essencial na avaliação e comparação de modelos de aprendizado de máquina, possibilitando a escolha de uma solução mais robusta para aplicações industriais.

4 CONCLUSÃO

A utilização de redes neurais convolucionais (CNNs) para a inspeção de qualidade demonstrou ser uma abordagem promissora, conforme discutido ao longo deste estudo. A crescente demanda por processos produtivos mais automatizados e precisos impulsiona a adoção de ferramentas baseadas em inteligência artificial (IA), permitindo a otimização da detecção de defeitos e minimizando falhas associadas a inspeções manuais. A análise realizada reforça a relevância desse tema para a engenharia química e a indústria em geral, especialmente no contexto da Indústria 4.0 (SCHMIDHUBER, 2015).

Os objetivos estabelecidos foram alcançados ao longo do estudo. Foi realizada uma revisão teórica sobre redes neurais convolucionais e sua aplicação na inspeção de qualidade industrial, além da implementação de diferentes modelos de CNN para a classificação de peças defeituosas e normais. Os modelos foram avaliados por meio de métricas como acurácia e matriz de confusão, o que permitiu identificar a configuração mais eficiente. Assim, a hipótese inicial de que a utilização de CNNs pode melhorar a automação na inspeção de qualidade industrial foi confirmada.

Os resultados obtidos demonstraram que a quantidade de camadas densas e o número de épocas de treinamento tiveram um impacto significativo no desempenho dos modelos testados. Dentre as arquiteturas avaliadas, o modelo correspondente ao Caso 4 foi o que apresentou os melhores resultados, destacando-se pela menor quantidade de falsos positivos na maioria dos testes realizados. Esse desempenho indica uma melhor adaptação do modelo aos padrões dos dados, reduzindo a ocorrência de classificações incorretas que poderiam comprometer a eficiência do processo industrial. A escolha do modelo quatro reforça a importância de um ajuste adequado dos hiperparâmetros, para alcançar um equilíbrio entre a capacidade preditiva e a robustez do modelo (RAWAT; WANG, 2017).

Dessa forma, a pesquisa reforça o potencial da visão computacional para a melhoria da inspeção de qualidade, alinhando-se às demandas contemporâneas de eficiência e automação.

4.1 Trabalho Futuros

Como continuidade deste estudo, sugere-se avaliar o desempenho de outros modelos de aprendizado profundo, bem como arquiteturas de CNN mais simples, capazes de operar em dispositivos com menor capacidade de memória, comuns na Indústria 4.0. Além disso, recomenda-se investigar a localização dos defeitos nas peças e realizar comparações entre os

resultados obtidos pela rede neural e a inspeção feita por operadores humanos. Outro ponto relevante seria a implementação de um sistema de inspeção em tempo real, por exemplo, em esteiras automatizadas, o que possibilitaria maior integração da visão computacional com os processos produtivos industriais.

Adicionalmente, propõe-se a análise de modelos mais simples de aprendizado profundo, como o *Perceptron* Multicamadas (MLP). Esse tipo de rede neural, embora menos especializado no processamento de imagens, pode ser uma alternativa viável quando associado a técnicas de extração de características, apresentando menor custo computacional e menor consumo de memória em comparação às CNNs. Dessa forma, sua aplicação seria especialmente interessante em contextos da Indústria 4.0, nos quais dispositivos embarcados e sistemas com restrições de hardware são amplamente utilizados. A comparação entre o MLP e a CNN permitiria verificar se a complexidade adicional da segunda se justifica em cenários industriais reais (RIEDMILLER; LERNEN, 2014).

REFERÊNCIAS

AGGARWAL, C. C. **Neural Networks and Deep Learning: A Textbook**. [S.l.]: Springer, 2018. ISBN 9783319944623. Citado na página 12.

Amazon Web Services. **O que é ajuste de hiperparâmetros?** 2024. <https://aws.amazon.com/pt/what-is/hyperparameter-tuning/>. Acessado em: 26 mar. 2025. Citado na página 27.

BHARADIYA, J. Convolutional neural networks for image classification. **International Journal of Innovative Science and Research Technology**, v. 8, n. 5, p. 673–677, 2023. Citado 2 vezes nas páginas 11 e 12.

BISHOP, C. M.; NASRABADI, N. M. **Pattern recognition and machine learning**. [S.l.]: Springer, 2006. v. 4. Citado na página 26.

CHEN, L. et al. Review of image classification algorithms based on convolutional neural networks. **Remote Sensing**, MDPI, v. 13, n. 22, p. 4712, 2021. Citado na página 11.

DABHI, R. **casting product image data for quality inspection**. 2020. Acessado em: 29 jan. 2025. Disponível em: <https://www.kaggle.com/datasets/ravirajsinh45/real-life-industrial-dataset-of-casting-product>. Citado na página 22.

DAVIDY, A. Thermodynamic design of organic rankine cycle (orc) based on petroleum coke combustion. **ChemEngineering**, MDPI, v. 5, n. 3, p. 37, 2021. Citado na página 18.

FERNANDEZ, A. et al. An introduction to the confusion matrix and its applications in machine learning. **Knowledge-Based Systems**, Elsevier, v. 150, p. 1–12, 2018. Citado na página 33.

FIELD, A. **Discovering statistics using IBM SPSS statistics**. [S.l.]: Sage publications limited, 2024. Citado na página 32.

GANESH, N. et al. Exploring deep learning methods for computer vision applications across multiple sectors: Challenges and future trends. **CMES-Computer Modeling in Engineering & Sciences**, Tech Science Press 871 CORONADO CENTER DR, SUTE 200, HENDERSON, NV 89052 USA, v. 139, n. 1, p. 103–141, 2024. Citado na página 10.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT Press, 2016. Disponível em: <https://www.deeplearningbook.org/>. Citado 3 vezes nas páginas 7, 25 e 26.

Google. **Google Colaboratory**. [S.l.], 2025. Acessado em: 12 fev. 2025. Disponível em: <https://colab.research.google.com/>. Citado na página 22.

HU, L. et al. Interpretable machine learning based on functional anova framework: algorithms and comparisons. **Applied Stochastic Models in Business and Industry**, Wiley Online Library, v. 41, n. 1, p. e2916, 2025. Citado na página 31.

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing in science & engineering**, IEEE Computer Society, v. 9, n. 03, p. 90–95, 2007. Citado na página 23.

KERAS. **Accuracy Metrics**. 2024. Disponível em: <https://keras.io/api/metrics/accuracy_metrics/>. Citado na página 26.

KERAS. **Adam Optimizer**. 2024. Disponível em: <https://keras.io/api/optimizers/adam/>. Citado na página 26.

KHANAM, R. et al. A comprehensive review of convolutional neural networks for defect detection in industrial applications. **IEEE Access**, IEEE, 2024. Citado na página 14.

KHAPARDE, Y. et al. Plant check: Potato leaf disease detection using cnn model. **International Journal of Engineering Applied Sciences and Technology**, v. 8, n. 5, p. 129–32, 2023. Citado 2 vezes nas páginas 20 e 21.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **International Conference on Learning Representations (ICLR)**, 2014. Disponível em: <https://arxiv.org/abs/1412.6980>. Citado na página 26.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2012. Citado na página 8.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, p. 436–444, 2015. Citado na página 7.

LI, C. et al. Fault diagnosis for distillation process based on cnn–dae. **Chinese Journal of Chemical Engineering**, Elsevier, v. 27, n. 3, p. 598–604, 2019. Citado 2 vezes nas páginas 18 e 19.

LIN, J. et al. Detection of a casting defect tracked by deep convolution neural network. **The International Journal of Advanced Manufacturing Technology**, Springer, v. 97, p. 573–581, 2018. Citado na página 22.

MONTGOMERY, D. C. **Design and analysis of experiments**. [S.l.]: John wiley & sons, 2017. Citado na página 32.

MURPHY, K. P. **Machine Learning: A Probabilistic Perspective**. MIT Press, 2012. Disponível em: <https://mitpress.mit.edu/9780262018029/machine-learning-a-probabilistic-perspective/>. Citado na página 26.

NAIR, G. E. H. V. Rectified linear units improve restricted boltzmann machines. **Proceedings of the 27th International Conference on Machine Learning (ICML)**, p. 807–814, 2010. Disponível em: <https://icml.cc/Conferences/2010/papers/432.pdf>. Citado na página 25.

O'DONOVAN, C.; GIANNETTI, C.; PLEYDELL-PEARCE, C. Revolutionising the sustainability of steel manufacturing using computer vision. **Procedia Computer Science**, Elsevier, v. 232, p. 1729–1738, 2024. Citado 2 vezes nas páginas 19 e 20.

PIMENTEL-GOMES, F. **Curso de estatística experimental**. [S.l.]: Digitaliza Conteúdo, 2023. Citado na página 32.

Python Software Foundation. **os — Miscellaneous operating system interfaces**. [S.l.], 2024. Acessado em: 12 fev. 2025. Disponível em: <https://docs.python.org/3/library/os.html>. Citado na página 23.

Python Software Foundation. **The Python Standard Library - JSON**. [S.l.], 2024. Acessado em: 12 fev. 2025. Disponível em: <https://docs.python.org/3/library/json.html>. Citado na página 23.

Python Software Foundation. **warnings — Warning control**. [S.l.], 2024. Acessado em: 12 fev. 2025. Disponível em: <https://docs.python.org/3/library/warnings.html>. Citado na página 23.

Python Software Foundation. **zipfile — Work with ZIP archives**. [S.l.], 2024. Acessado em: 12 fev. 2025. Disponível em: <https://docs.python.org/3/library/zipfile.html>. Citado na página 23.

RAWAT, W.; WANG, Z. Deep convolutional neural networks for image classification: A comprehensive review. **Neural Computation**, MIT Press, v. 29, n. 9, p. 2352–2449, 2017. Citado na página 37.

RIEDMILLER, M.; LERNEN, A. Multi layer perceptron. **Machine learning lab special lecture, University of Freiburg**, v. 24, p. 11–60, 2014. Citado na página 38.

SAI, T. V.; VINOD, T.; SOWMYA, G. A critical review on casting types and defects. **Engineering and Technology**, v. 3, n. 2, p. 463–468, 2017. Citado na página 22.

SALEHIN, I.; KANG, D.-K. A review on dropout regularization approaches for deep neural networks within the scholarly domain. **Electronics**, MDPI, v. 12, n. 14, p. 3106, 2023. Citado 2 vezes nas páginas 15 e 16.

SCHMIDHUBER, J. **Deep Learning in Neural Networks: An Overview**. [S.l.]: Elsevier, 2015. v. 61. 85–117 p. Citado na página 37.

SIMONYAN, A. Z. K. Very deep convolutional networks for large-scale image recognition. **International Conference on Learning Representations (ICLR)**, 2015. Disponível em: <https://arxiv.org/pdf/1409.1556.pdf>. Citado na página 25.

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. **Information processing & management**, Elsevier, v. 45, n. 4, p. 427–437, 2009. Citado na página 32.

SOUSA, W. R. N. d. Criação e avaliação de modelos de prognóstico futuro de linhas de costa, utilizando regressão estatística e redes neurais artificiais, a partir das séries temporais de imagens de satélite. 2022. Citado 2 vezes nas páginas 9 e 16.

SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. **The journal of machine learning research**, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014. Citado na página 26.

TAYE, M. M. Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions. **Computation**, v. 11, n. 3, 2023. ISSN 2079-3197. Disponível em: <https://www.mdpi.com/2079-3197/11/3/52>. Citado na página 13.

TEAM, T. pandas D. **pandas: powerful Python data analysis toolkit**. [S.l.], 2024. Acessado em: 12 fev. 2025. Disponível em: <https://pandas.pydata.org/docs/>. Citado na página 24.

TensorFlow Developers. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. [S.l.], 2024. Acessado em: 12 fev. 2025. Disponível em: <https://www.tensorflow.org/>. Citado na página 23.

THEISEN, M. F. et al. Digitization of chemical process flow diagrams using deep convolutional neural networks. **Digital Chemical Engineering**, Elsevier, v. 6, p. 100072, 2023. Citado na página 19.

WASKOM, M. L. Seaborn: statistical data visualization. **Journal of Open Source Software**, v. 6, n. 60, p. 3021, 2021. Citado na página 23.

WU, T.-W. et al. Applications of convolutional neural networks for intelligent waste identification and recycling: A review. **Resources, Conservation and Recycling**, Elsevier, v. 190, p. 106813, 2023. Citado na página 9.

ZAFAR, A. et al. A comparison of pooling methods for convolutional neural networks. **Applied Sciences**, MDPI, v. 12, n. 17, p. 8643, 2022. Citado 2 vezes nas páginas 14 e 15.

ZHANG, A. et al. **Dive into Deep Learning**. Aston Zhang, 2020. Disponível em: <https://pt.d2l.ai/chapter_convolutional-neural-networks/conv-layer.html>. Citado na página 27.

ZHAO, X. et al. A review of convolutional neural networks in computer vision. **Artificial Intelligence Review**, Springer, v. 57, n. 4, p. 99, 2024. Citado na página 12.

ZHONG, S. et al. Molecular image-convolutional neural network (cnn) assisted qsar models for predicting contaminant reactivity toward oh radicals: Transfer learning, data augmentation and model interpretation. **Chemical Engineering Journal**, Elsevier, v. 408, p. 127998, 2021. Citado na página 20.