INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
CAMPUS BETIM

ISRAEL MONTEIRO DIAS

SELEÇÃO AUTOMÁTICA DE PEÇAS UTILIZANDO VISÃO COMPUTACIONAL

BETIM
2025

Israel Monteiro Dias

SELEÇÃO AUTOMATICA DE PEÇAS UTILIZANDO VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais *Campus* Betim, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Controle e Automação.

ORIENTADORA: Michelle Mendes Santos

Betim

2025

FICHA CATALOGRÁFICA

D541s Dias, Israel Monteiro

Seleção automática de peças utilizando visão computacional / Israel Monteiro Dias. – 2025.

47 f. : il.

Trabalho de conclusão de curso (Bacharelado em Engenharia de Controle e Automação) - Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Câmpus Betim, 2025.

Orientação: Prof. Ma. Michelle Mendes Santos

1. Visão computacional. 2. Raspberry Pi. 3. Redes neurais (Computação). 4. Engenharia de Controle e Automação. I Dias, Israel Monteiro. II. Título.

CDU: 519.6

Ficha catalográfica elaborada pelo Bibliotecário Denísio Pereira Marcos CRB-6/3142

Israel Monteiro Dias

SELEÇÃO AUTOMÁTICA DE PEÇAS UTILIZANDO VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais *Campus* Betim, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Aprovado em: 06/08/2025 pela banca examinadora:

Michelle Mendes Santos

Prof ª. Me. Michelle Mendes Santos (Orientadora) – IFMG Campus Betim

Prof. Me. Virgil Del Duca Almeida – IFMG Campus Betim

Eng. Robson Macedo dos Santos

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema automatizado para inspeção e classificação de peças industriais utilizando visão computacional. O objetivo é substituir ou complementar a inspeção manual, que apresenta limitações como erros. O sistema foi implementado em uma plataforma embarcada Raspberry Pi, integrando uma câmera para captura de imagens e a arquitetura YOLOv8 para detecção e classificação das peças como boas ou ruins. O processamento das imagens foi realizado com a biblioteca OpenCV, e os resultados são exibidos por meio de uma interface homem-máquina (IHM) desenvolvida em Python. O modelo foi treinado com imagens anotadas manualmente e demonstrou bom desempenho em métricas como precisão, revocação e mAP. Os testes indicam que a solução é viável para aplicações industriais, oferecendo uma alternativa de baixo custo, eficiente e confiável para o controle de qualidade automatizado.

Palavras-chave: Visão computacional; Raspberry Pi; classificação; Redes neurais convolucionais

ABSTRACT

This work presents the development of an automated system for the inspection and classification of industrial parts using computer vision. The objective is to replace or complement manual inspection, which presents limitations such as human errors and low repeatability. The system was implemented on a Raspberry Pi embedded platform, integrating a camera for image capture and the YOLOv8 architecture for object detection and classification of parts as good or defective. Image processing was performed using the OpenCV library, and the results are displayed through a human-machine interface (HMI) developed in Python. The model was trained with manually annotated images and showed good performance in metrics such as precision, recall, and mAP. Tests indicate that the proposed solution is feasible for industrial applications, offering a low-cost, efficient, and reliable alternative for automated quality control.

Keywords: Computer vision; Raspberry Pi; Part classification; YOLOv8; Convolutional neural networks.

LISTA DE ABREVIATURAS E SIGLAS

CNN	Convolutional Neural Network (Rede Neural Convolucional)
FPS	Frames Per Second (Quadros por Segundo)
GPU	Graphics Processing Unit (Unidade de Processamento Gráfico)
HMI / IHM	Human-Machine Interface / Interface Homem-Máquina
IFMG	Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais
IoU	Intersection over Union (Interseção sobre União) [opcional, caso citada em outras partes]
mAP	Mean Average Precision (Precisão Média Média)
OpenCV	Open Source Computer Vision Library
ReLU	Rectified Linear Unit (Unidade Linear Retificada)
RGB	Red, Green, Blue (vermelho, verde, azul)
SVM	Support Vector Machine (Máquina de Vetores de Suporte) [caso mencionada na comparação]
TCC	Trabalho de Conclusão de Curso
YOLO	You Only Look Once (Você Só Olha Uma Vez)

SUMÁRIO

1 INTRODUÇÃO

Em processos industriais, a uma grande variedade peças isto é muito importante para competitividade econômica e variedade de produtos do seu portfólio. Essa diversidade abrange componentes com diferentes tamanhos, formatos, materiais, cores e funções, especialmente em setores como o automotivo. De acordo com Aghion e Howitt (1992), a capacidade de inovar e diversificar a produção é determinante para manter a competitividade, permitindo que novas soluções e produtos substituam tecnologias e métodos anteriores por alternativas mais eficientes.

Entretanto quando falamos em seleção, essa multiplicidade pode se mostrar desafiadora, especialmente feito de forma manual O controle de qualidade nesse contexto exige conhecimento técnico sobre características específicas, bem como a atuação de operadores treinados, capazes de identificar padrões visuais e dimensionais com precisão, isto demanda atenção constante e alto nível de concentração. Segundo Denis (2007), “a classificação consiste em organizar um produto por tamanho, cor, formato e qualidade, a fim de formar um grupo”. Contudo, esse tipo de tarefa está sujeita a limitações humanas, fatores que comprometem tanto a padronização da qualidade quanto a produtividade da linha de produção.

Diante dessas limitações inerentes à inspeção manual, a automação surge como uma alternativa promissora para aumentar a eficiência e a consistência do controle de qualidade. Sistemas baseados em visão computacional e inteligência artificial têm sido cada vez mais empregados para minimizar erros humanos, acelerar o processo de seleção e garantir maior uniformidade na classificação.

A Figura 1 ilustra um exemplo da complexidade envolvida na variedade de peças automotivas, mostrando um veículo HB20 completamente desmontado. É possível observar a grande quantidade de componentes distintos que compõem o sistema veicular, o que evidencia o quão complexo pode ser o processo de inspeção e classificação em ambientes industriais.

Figura 1 – Peças de um automóvel Hyundai HB20

Fonte: Pedro Rubens/ Revista Quatro Rodas

Existem diversos métodos e técnicas de inspeção, tais como o método pulso-eco, método por transparência, método por ressonância, técnica por contato, técnica por imersão, inspeção manual e inspeção automática (More; Jesús Domech, 2004).

Os sistemas de inspeção automatizada podem ser baseados em aplicações da tecnologia computacional, que atualmente desempenha papel fundamental na busca pela competitividade industrial (GOES; MIKOS, 2014). Por meio de dispositivos dotados de visão artificial, a indústria é capaz de realizar tarefas de automação em processos que minimizam erros e que são guiados pelas normas de qualidade já consolidadas pelas organizações e clientes.

Além de contribuir para a automação de tarefas repetitivas, os sistemas de visão computacional são amplamente utilizados em processos de inspeção visual, permitindo a detecção de falhas, anomalias ou desvios dimensionais em peças industriais. Equipados com câmeras e algoritmos de processamento de imagem, esses sistemas analisam características visuais como formato, integridade, presença de defeitos e acabamento superficial, garantindo que as peças atendam aos padrões de qualidade exigidos. Dessa forma, a tecnologia permite uma análise objetiva e padronizada, minimizando a subjetividade comum na inspeção manual.

O presente trabalho propôs a implementação de uma metodologia computacional para a análise e seleção automática de peças automotivas metálicas, com o objetivo de classificá-las segundo suas tipologias e critérios de qualidade preestabelecidos. Para isso, utilizou-se a tecnologia de visão computacional integrada

a um sistema de detecção baseado em redes neurais convolucionais, permitindo a identificação de defeitos e variações estruturais de forma autônoma. Essa abordagem visa substituir ou complementar a inspeção manual, reduzindo erros humanos e promovendo maior padronização e eficiência nos processos industriais.

1.1 Justificativa

A análise histórica demonstra que falhas relacionadas à qualidade de peças produzidas podem estar associadas a diversos incidentes, desde perdas financeiras até riscos fatais à vida de consumidores finais. A literatura apresenta inúmeros exemplos de acidentes significativos causados por falhas nos processos de inspeção. Um exemplo marcante são as trincas por corrosão intergranular sob tensão que não foram identificadas durante uma inspeção por ultrassom na usina nuclear de Nine Mile Point, em 1981, ocasionando impactos severos na indústria nuclear (ALBUQUERQUE , 2011).

Sabe-se que esses incidentes podem ser auditados e prevenidos por meio de processos eficazes de coleta e validação de dados. Durante paradas de manutenção, construção de novas unidades ou grandes modificações de projeto, diversos ensaios são realizados com o objetivo de garantir a qualidade dos serviços executados e avaliar a integridade estrutural dos componentes.

Entretanto, essas falhas de qualidade acarretam tarefas adicionais à sequência de atividades operacionais, exigindo dos profissionais níveis elevados de produtividade. Como consequência, muitos desses profissionais operam sob constante pressão, devido à carga de trabalho intensa, o que pode comprometer a qualidade dos ensaios executados.

Na tentativa de atender às demandas de produtividade, é comum que os profissionais enfrentem o dilema entre manter a qualidade do serviço e cumprir prazos apertados. Conforme destacado por Albuquerque (2011), o equilíbrio dessa delicada relação depende, cada vez mais, da intervenção de mecanismos externos capazes de

neutralizar a pressão produtiva, tornando necessária a adoção de acompanhamentos especializados, devidamente documentados, auditáveis e controláveis.

1.2 Objetivos

O objetivo principal deste trabalho é implementar um sistema de visão computacional capaz de auxiliar na análise da confiabilidade humana em processos de inspeção e seleção de peças.

Os objetivos específicos são:

- Compreender se os dados coletados são compatíveis com a robustez do hardware utilizado;
- Dimensionar os elementos do sistema de acordo com a estrutura física analisada;
- Analisar os resultados obtidos a partir de testes realizados fora do ambiente do protótipo;
- Avaliar se, com a aplicação do sistema proposto, será possível automatizar a análise de qualidade e a tipologia das peças, comparando os resultados com os de um colaborador experiente e habilitado;

1.3 Organização do Trabalho

No Capítulo 2, serão abordados temas relacionados à obtenção e ao processamento de imagens, bem como ao uso dessas imagens em redes neurais e às tecnologias utilizadas no desenvolvimento do projeto. O Capítulo 3 refere-se à metodologia adotada, descrevendo o processo de desenvolvimento do sistema e os materiais e recursos empregados. No Capítulo 4, são apresentados os resultados obtidos a partir dos testes realizados, incluindo métricas de desempenho e análise crítica do comportamento do sistema em ambiente industrial. Por fim, o Capítulo 5 apresenta as conclusões do trabalho, destacando as contribuições alcançadas, as limitações identificadas e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TÉORICA

A seleção automática de peças por meio de visão computacional envolve um conjunto de etapas interligadas que formam um ciclo contínuo e sistemático. Inicialmente, o processo tem início com a aquisição das imagens das peças a serem analisadas, etapa fundamental para garantir a qualidade dos dados que serão utilizados nas fases subsequentes. Essas imagens capturadas passam por um processamento prévio, que inclui técnicas de filtragem e realce, com o objetivo de preparar o material visual para a análise mais aprofundada.

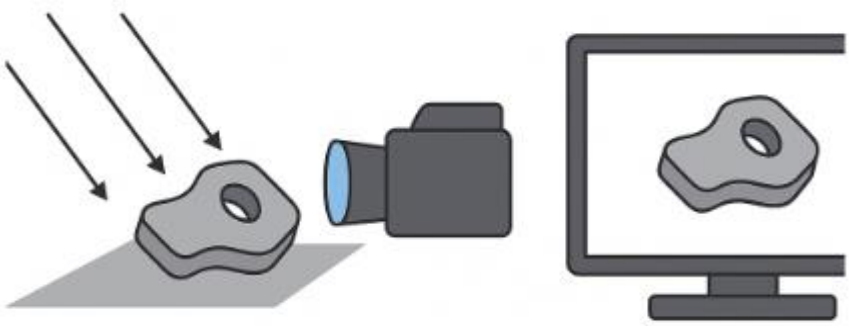
Em seguida, as imagens tratadas são submetidas a algoritmos baseados em redes neurais, que desempenham o papel de reconhecer e classificar as características das peças, determinando se estão em condições adequadas ou apresentam algum tipo de defeito. Para que esses algoritmos possam operar de forma eficaz, é necessário que um modelo seja previamente treinado, testado e validado utilizando conjuntos de dados representativos, assegurando a sua robustez e precisão na identificação dos diferentes padrões.

Dessa forma, a visão computacional associada ao aprendizado de máquina propicia a automação do processo de seleção, reduzindo a necessidade de inspeção manual e aumentando a confiabilidade e velocidade na triagem das peças.

2.1 Aquisição de Imagem

O processo de aquisição de imagem consiste na captura de um objeto tridimensional e sua conversão em um modelo bidimensional. Em termos práticos, a câmera utiliza seus fotorreceptores para captar a luz visível refletida pelo ambiente e projetá-la em uma superfície 2D. Este processo é ilustrado pela figura 2.

Figura 2 – Representação processo de aquisição de imagem



Fonte: Elaborado pelo autor.

Segundo Gonzalez e Woods (2018), o processo de aquisição de imagem envolve a captura da luz refletida por objetos tridimensionais, convertendo-a em um modelo bidimensional.

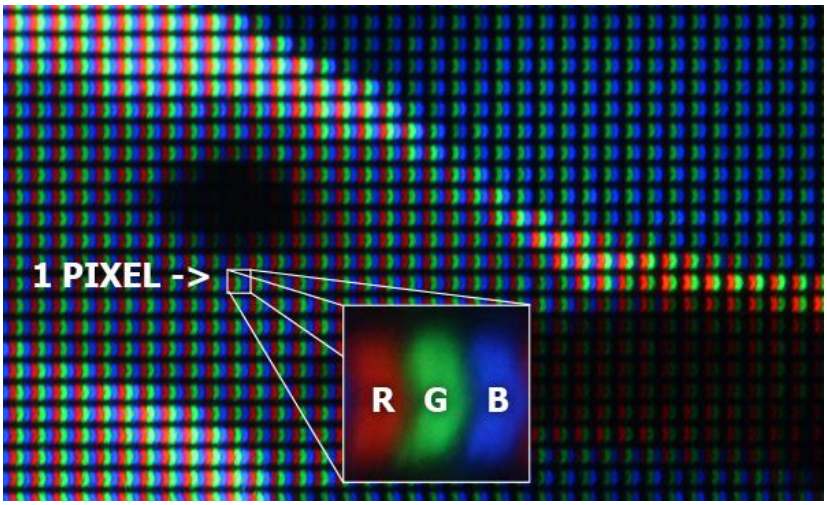
Em contraponto ao processo bidimensional, Tsai (1987) destaca que, no campo da visão artificial, existem câmeras unidimensionais (também chamadas de câmeras de varredura linear) e câmeras bidimensionais (de área), sendo que estas últimas são as mais amplamente utilizadas nas aplicações industriais.

Dentro das categorias de câmeras disponíveis, diversas escolhas técnicas devem ser feitas, como, por exemplo, a definição entre imagens monocromáticas ou coloridas, o ângulo e o campo de visão, a resolução, entre outras características relevantes para a aplicação pretendida.

A imagem formada a partir da amostragem e da quantização de uma função contínua, como processo de aquisição de imagens bidimensionais. Nesse contexto, a imagem 2D é convertida em uma matriz de valores, onde cada elemento corresponde a um ponto específico no plano. Cada um desses pontos é chamado de pixel e representa a menor porção de informação da imagem, responsável por codificar a cor e

a intensidade da luz naquele ponto específico (Gonzales; Woods, 2018). A imagem digital pode ser codificada em diferentes representações de cor, mas muitas vezes é convertida para o modelo RGB para que possa ser exibida e interpretada adequadamente pelo olho humano, uma vez que o RGB corresponde ao funcionamento fisiológico da visão humana e aos dispositivos de exibição comuns, como monitores e telas. A Figura 3 exemplifica bem esse conceito ao apresentar a imagem de um pássaro ampliada evidenciando a matriz de pixels, cada um com diferentes intensidades de cor.

Figura 3 – imagem de um pássaro ampliada



Fonte: https://apenasimagens.com/pt/pixel-imagem-digital/.

2.2 Processamento e filtro de imagem

Existem componentes em uma imagem que, embora importantes para representar fielmente o mundo real, podem aumentar significativamente o volume de processamento e dificultar a separação das estruturas presentes na imagem. Por isso, o processamento de imagem tem como objetivo transformar a imagem capturada em uma representação computacionalmente mais compacta, reduzindo interferências causadas por ruídos e eliminando informações redundantes. Essa etapa é essencial para tornar a imagem mais adequada às operações de análise automatizada, como detecção de bordas, segmentação de objetos ou extração de características. Segundo Crosta (1999), "o objetivo principal do processamento de imagens é o de remover essas

barreiras, inerentes ao sistema visual humano, facilitando a extração de informações a partir de imagens."

Entre as técnicas mais utilizadas nesse processo, destaca-se a conversão da imagem para escala de cinza, que facilita a aplicação de filtros e algoritmos sem depender da informação de cor original.

A perspectiva digital pode ser entendida como uma função bidimensional f(x,y), que x e y indicam posições no espaço e o valor de f em cada coordenada (x,y) representa as variedade de cor correspondente àquele ponto da imagem. Quando as posições x e y e os valores de intensidade de f são finitos e discretos, tem-se uma imagem digital. Nesse formato, a imagem é formada por um número limitado de elementos, chamados pixels, sendo que cada pixel possui um valor numérico e uma localização bem definida (ACHARYA; RAY, 2005).

Em termos numéricos, essa intensidade geralmente é expressa por valores inteiros que variam de 0 (preto absoluto) a 255 (branco absoluto) em imagens de 8 bits, sendo que valores intermediários correspondem a diferentes níveis de cinza. Essa simplificação elimina as componentes cromáticas presentes em imagens coloridas, reduzindo a quantidade de dados a serem processados e, consequentemente, diminuindo o custo computacional. (PEDRINI; SCHWARTZ, 2008).

Um outro exemplo de processamento de imagens é a utilização do operador de Sobel, um filtro amplamente empregado em processamento de imagens para identificar mudanças na intensidade luminosa. Esse operador trabalha com aproximações matemáticas conhecidas como gradientes, que indicam os pontos de maior variação ou crescimento na intensidade da imagem, destacando assim as bordas presentes (Sonka; Hlavac; Boyle, 2014). A Figura 4 ilustra essa aplicação, onde os contornos se tornam mais visíveis após o processamento.

Segundo Silva e Alves (2008), "o detector de bordas de Sobel tem como objetivo destacar bordas de uma imagem no sentido horizontal e vertical. Está baseado na

aproximação de Sobel para a derivada, retornando a borda onde o gradiente é máximo."

Figura 4 – Operador Sobel

Fonte: Fiveko, 2023.

Além do operador Sobel, existem diversas outras técnicas e operadores capazes de destacar e realçar bordas em imagens. Esses métodos são geralmente classificados como operações de gradiente, pois destacam regiões com maior variação de intensidade

Outra abordagem complementar ao realce de bordas é a segmentação de imagem, cujo objetivo é identificar e isolar regiões de interesse. Esse processo pode ser realizado de forma automática, por meio do agrupamento ou divisão de áreas da imagem com base em características semelhantes entre os pixels, tais como cor, textura ou forma. (GONZALEZ; WOODS, 2018).

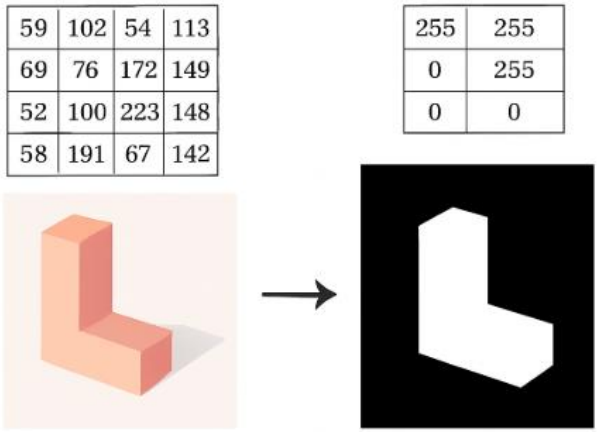
Na segmentação, a imagem é dividida em partes distintas, chamadas regiões ou segmentos, que apresentam propriedades visuais similares. O propósito é agrupar pixels que pertencem ao mesmo objeto ou área, separando-os do fundo ou de outras regiões da cena. Esse agrupamento pode considerar atributos como cor, textura e forma, permitindo distinguir elementos relevantes para a análise.

Essa forma de processamento de imagem pode reduzir consideravelmente a quantidade de dados na matriz de pixels que será analisada pelo algoritmo, diminuindo assim o custo computacional da operação. Dessa forma, o sistema torna-se mais

eficiente ao lidar com grandes volumes de imagens ou em aplicações em tempo real (GONZALEZ, 2009).

A Figura 3 ajuda a demonstrar, de forma didática, como técnicas como filtragem, escala de cinza e segmentação diminuem a complexidade dos dados que o algoritmo precisa interpretar.

Figura 3 – Redução da matriz de pixels após o processamento de imagem



Fonte: Elaborado pelo autor.

2.3 Aprendizado Computacional de Decisão

O Aprendizado de Máquina se da pela construção de algoritmos que melhorem seu desempenho por meio de exemplos. Para isso é necessário o uso informações positivas e negativas para gerar o conhecimento do computador, que são hipóteses geradas a partir dos dados (Ludermir, 2021).

De acordo com Ludermir (2021), As técnicas de aprendizado de máquinas são orientadas a dados, isto é, aprendem automaticamente a partir de grandes volumes de dados. Os algoritmos de AM geram hipóteses a partir dos dados.

Resolver problemas por meio do Aprendizado de Máquina nem sempre é uma tarefa simples, pois exige o cumprimento de certos requisitos e a utilização de um conjunto de dados adequado. A qualidade dos dados empregada pode impactar diretamente o desempenho e a eficácia do modelo produzido (Zhang, 2023).

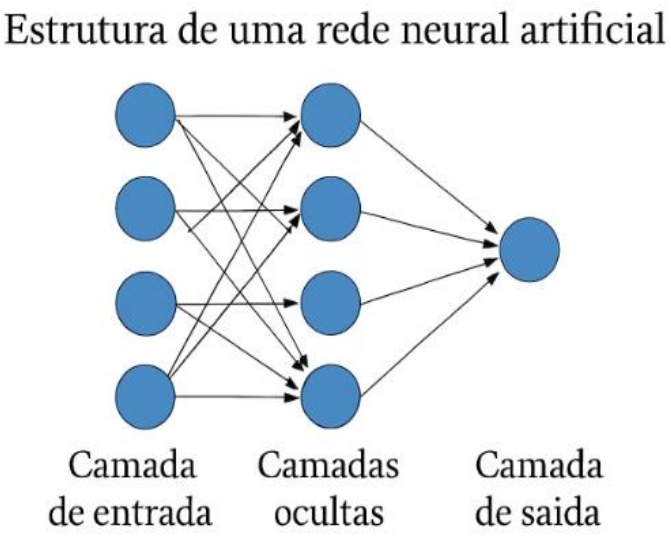
Uma forma importante de organizar o Aprendizado de Máquina são as redes neurais artificiais, que simulam o funcionamento do cérebro humano para aprender e

Identificar padrões complexos nos dados. Essas redes dependem fortemente da qualidade dos dados para que o aprendizado seja eficiente e os resultados sejam satisfatórios (Liu, 2011).

Diversos estudos em inteligência artificial baseiam-se na estrutura neurológica do cérebro humano para tornar possível o processo de aprendizado computacional. Um dos principais modelos desenvolvidos a partir dessa inspiração é a rede neural artificial, que busca simular a capacidade humana de tomada de decisão e adaptação a situações adversas.

Nessas redes, os neurônios artificiais apresentam funcionamento semelhante ao modelo do perceptron, sendo organizados em camadas interconectadas que compõem a estrutura da rede. Essa arquitetura pode ser observada na Figura 5, onde são representadas as camadas de entrada, ocultas e de saída, fundamentais para o funcionamento da rede neural.

Figura 5 - Modelo de rede neural simples



Fonte: Adaptado de Goodfellow et al., 2016.

No contexto das redes neurais artificiais, cada neurônio artificial, também conhecido como perceptron, é responsável por processar as informações recebidas das camadas anteriores. Essas informações chegam por meio de entradas, cada uma associada a um peso, que representa a força sináptica da conexão analogamente ao que ocorre nas sinapses dos neurônios biológicos (Zhang, 2023).

O neurônio realiza então uma soma ponderada dessas entradas e, com base nesse valor, é calculado o seu nível de ativação. Caso esse nível ultrapasse um determinado limiar, o neurônio gera uma saída, que será transmitida à próxima camada da rede. Esse modelo constitui a estrutura básica e amplamente adotada para representar o comportamento de um neurônio humano em sistemas computacionais.

Após o recebimento dos dados de entrada, eles são processados pelos perceptrons das camadas ocultas, que realizam a multiplicação dos valores recebidos pelos respectivos pesos sinápticos. O resultado dessa operação é então transmitido para os neurônios da próxima camada, seguindo o fluxo da rede até atingir a camada de saída.

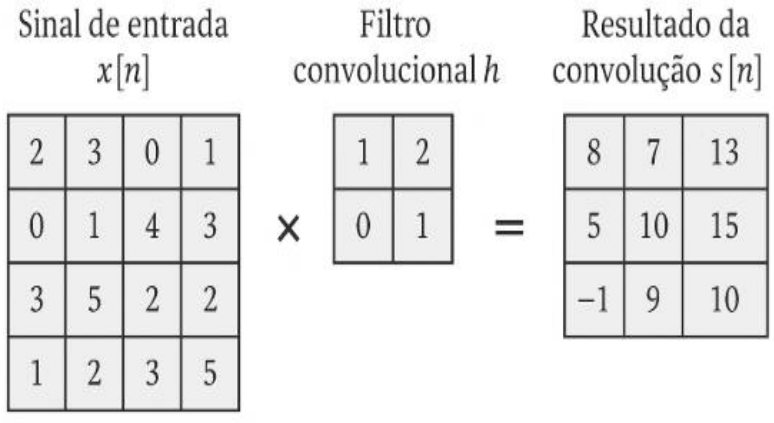
Quando interligados em múltiplas camadas, os neurônios artificiais apresentam um grande potencial para mapear relações não lineares entre as entradas e saídas, o que os torna extremamente eficazes em tarefas complexas de classificação, reconhecimento de padrões e tomada de decisão (Zhang, 2023).

2.3.1 Redes Neurais Convolucionais (CNN)

As redes neurais convolucionais (CNNs) são amplamente utilizadas em aplicações como visão computacional, detecção de falhas em sistemas robóticos e análise de imagens em ambientes industriais. Essas redes são especialmente eficazes para o reconhecimento e a classificação automática de objetos em imagens, devido à sua capacidade de extrair e aprender características espaciais e hierárquicas dos dados de entrada.

O comportamento de um CNN pode ser observado a partir do processo de convolução, em que uma matriz de entrada, representando a imagem, é processada por filtros convolucionais que percorrem a imagem extraindo padrões específicos, como bordas, texturas e formas. Esse procedimento gera mapas de características que destacam regiões relevantes da imagem para a tarefa de detecção ou classificação. (Goodfellow, 2016). Afigura 6 e um Exemplo de convolução bidimensional entre um sinal de entrada x[n] e um filtro convolucional h, resultando no mapa de saída s[n]. O processo percorre a matriz de entrada aplicando o filtro sobre regiões locais e somando os produtos para compor a saída.

Figura 6 - Representação da estrutura de uma rede neural convolucional (CNN



Fonte: Adaptado de Goodfellow et al. (2016).

2.3.2 Detecção de Objetos com YOLO e DarKnet

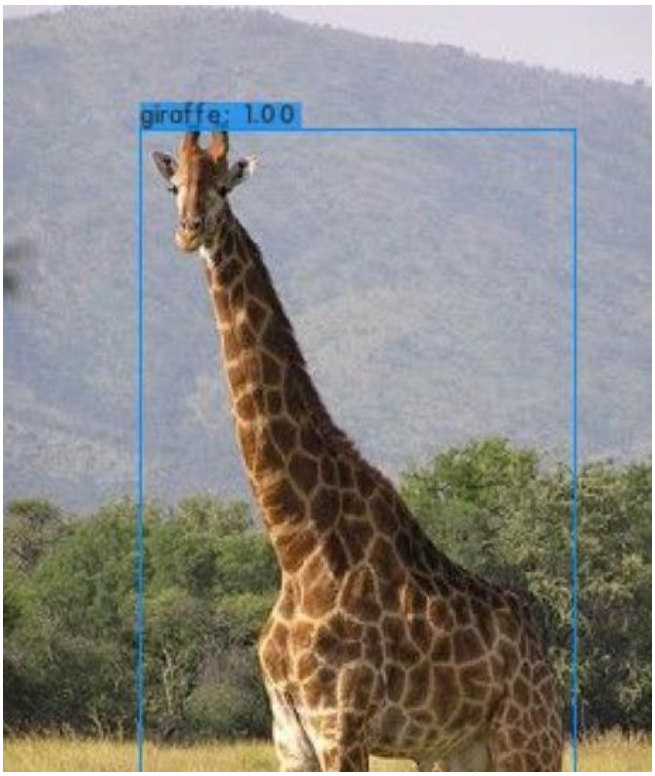
Dando continuidade à aplicação prática das redes neurais convolucionais (CNNs), destaca-se o modelo YOLO (*You Only Look Once*) como uma das arquiteturas mais eficientes para a detecção de objetos em tempo real. Enquanto as CNNs tradicionais são amplamente utilizadas para classificação de imagens, o YOLO expande essa abordagem ao integrar simultaneamente a localização e a identificação de múltiplos objetos em uma única análise (Redmon, 2016).

Essa arquitetura é implementada com frequência no framework Darknet, uma plataforma de código aberto desenvolvida especificamente para o YOLO. Escrito em C e CUDA, o Darknet permite a execução otimizada dos modelos tanto em CPUs quanto em GPUs, sendo ideal para aplicações com requisitos de alta performance e processamento em tempo real. Seu diferencial está na leveza, portabilidade e ampla compatibilidade com diferentes versões do YOLO. O projeto conta com uma comunidade ativa, o que favorece melhorias contínuas e facilidade de integração com bibliotecas de alto nível, como PyTorch e Ultralytics, permitindo ainda a implementação embarcada em dispositivos como o Raspberry Pi (LIN ,2014).

O YOLO analisa a imagem em uma única passagem, dividindo-a em uma grade e, para cada região, prevê caixas de delimitação (*bounding boxes*) e probabilidades de classe. A figura 7 comprova este funcionamento onde há diversos objetos dentro da

imagem porem o algoritmo reconhece somente a classe a qual foi treinada reconhecendo assim uma girafa e mostrando uma retângulo azul indicando a decisão.

Figura 7– Reconhecimento de imagens utilizando yolo



Fonte: Elaborado pelo autor.

Isso permite a identificação precisa e rápida de objetos, mesmo em cenas complexas. Um exemplo de aplicação comum são os sistemas de monitoramento por câmeras de trânsito, onde há a necessidade de identificar diversos elementos simultaneamente, como veículos, placas e pedestres.

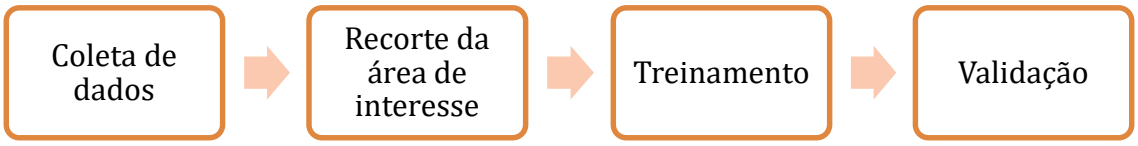
Ao contrário das abordagens anteriores voltadas apenas para classificação, o YOLO é projetado especificamente para detecção dinâmica, sendo baseado em técnicas de aprendizado de máquina e visão computacional (Redmon, 2016). Sua implementação é frequentemente realizada em Python, utilizando bibliotecas como o OpenCV, que fornece ferramentas para leitura, manipulação e exibição de imagens em tempo real.O uso de modelos como YOLO tem revolucionado o controle de qualidade e a automação industrial, permitindo a inspeção rápida e precisa de peças, reduzindo

custos operacionais e aumentando a confiabilidade dos processos produtivos (Bradski, 2000).

3 METODOLOGIA

Levando em consideração os objetivos definidos para este trabalho, foi necessário estabelecer uma sequência de processos técnicos que possibilitassem a implementação e compreensão do sistema de seleção automática de peças. O desenvolvimento de uma solução baseada em visão computacional para aplicação em ambientes industriais exigiu o planejamento de etapas tanto sequenciais quanto paralelas, envolvendo conhecimentos das áreas de automação, eletrônica embarcada e inteligência artificial. A estruturação do sistema proposto demandou a integração de diversas tecnologias, desde a aquisição e anotação das imagens até o treinamento da rede neural convolucional YOLOv8 e a implementação em um sistema embarcado com Raspberry Pi. A Figura 8 apresenta uma visão geral do fluxo de desenvolvimento adotado neste projeto.

Figura 8 – Fluxo de desenvolvimento



Fonte: Elaborado pelo autor

3.1 Coleta de dados

Para a composição do banco de imagens, foram capturadas 200 fotografias de caixas de marcha, de forma a simular o ambiente inspeção. Para a coleta, utilizou-se o módulo de câmera OV5647 (Figura 9), que possui resolução de 5 megapixels e capacidade de gravação, custo, tamanho e qualidade de captura. Esse módulo foi acoplado a um Raspberry Pi 4B (Figura 10), um minicomputador de baixo custo, equipado com processador quad-core, memória RAM 4GB e interfaces de comunicação integradas.

Figura 9– Módulo de câmera OV5647

Fonte: Elaborado pelo autor

Figura 10 – Raspberry Pi 4B

Fonte: Elaborado pelo autor

Para garantir estabilidade e robustez mecânica, foi projetada e construída uma case personalizada em impressora 3D, destinada a abrigar o Raspberry Pi e o módulo

de câmera. A estrutura mantém o posicionamento fixo dos componentes e oferece proteção contra névoa industrial, poeira e vibrações, além de possibilitar o ajuste do ângulo da câmera, garantindo maior flexibilidade para adequação do enquadramento conforme a necessidade do ambiente. Como pode ser visto na Figura 11 câmera e o módulo estão montados dentro da case.

Figura 11 – Hardware (câmera e raspberry montados na case)

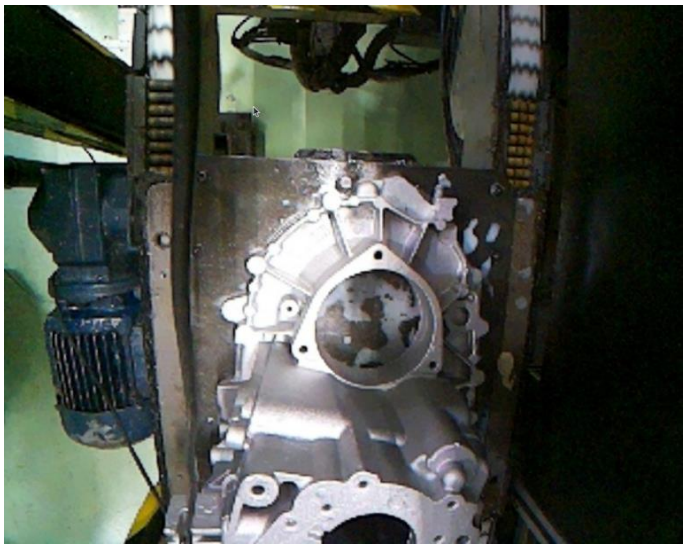
Fonte: Elaborado pelo autor

O hardware foi instalado em uma estrutura fixa, posicionada estrategicamente acima da linha de movimentação das peças, de modo a garantir uma visão clara e contínua da área de interesse as caixas de macha e suportes foram movimentados sob a câmera, tanto manualmente quanto por meio de dispositivos automatizados integrado ao ambiente industrial, simulando as condições reais de transporte e manuseio das peças no processo produtivo.

As Figuras 11 e 12 ilustram essa configuração durante a etapa de aquisição de imagens nesta imagens pode ser observado que foram realizadas variações intencionais nas condições de iluminação com o objetivo de aumentar a robustez dos dados obtidos. Assim, o banco de dados resultante reflete uma diversidade de

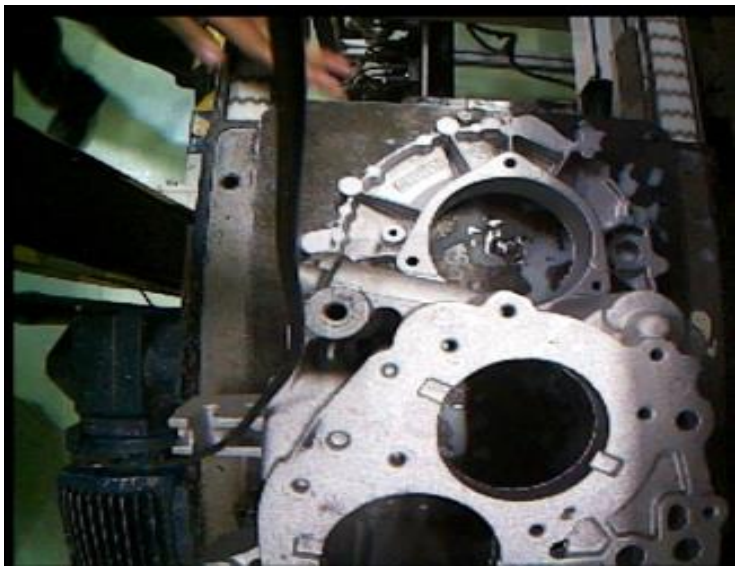
situações reais, contribuindo para a melhora da generalização e da precisão do modelo treinado.

Figura 12 – Imagem com iluminação artificial



Fonte: Elaborado pelo autor.

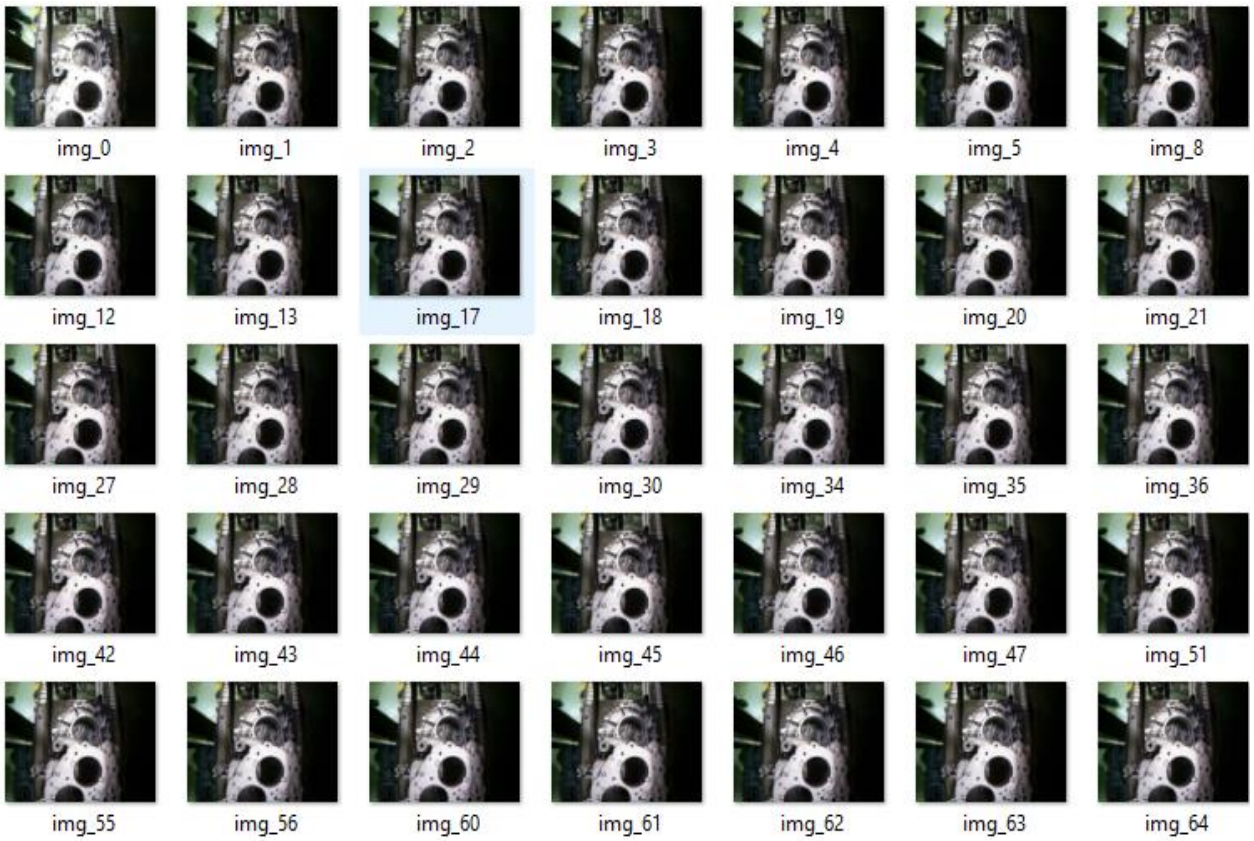
Figura 13 – imagem sem iluminação artificial.



Fonte: Elaborado pelo autor.

Ao final deste processo, foi obtido um banco de dados abrangente e representativo, contendo imagens capturadas sob diversas condições ambientais e operacionais, garantindo uma base sólida para o treinamento do modelo de visão computacional, como pode ser visto na Figura 14, que apresenta diversas imagens de caixas de marcha com iluminação artificial.

Figura 14 – Banco imagens com peças iluminação artificial.



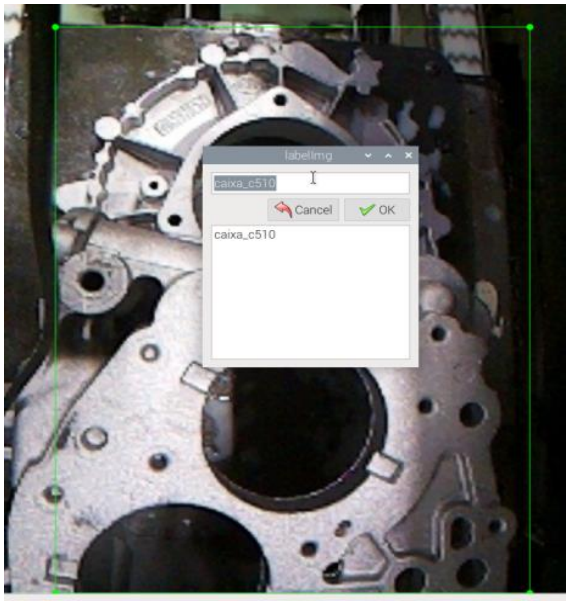
Fonte: Elaborado pelo autor.

3.2 Recorte da área de interesse

O recorte da área de interesse consiste na seleção e isolamento da região da imagem que contém pontos específicos da peça metálica, eliminando partes desnecessárias do cenário. Essa abordagem permite focar o processamento e a análise apenas nos detalhes relevantes, reduzindo ruído e aumentando a precisão do algoritmo de detecção e classificação Foi formulado um conjunto contendo 100 imagens, cuidadosamente selecionadas e anotadas, que serviram como base para o

treinamento e validação do modelo de visão computacional.A Figura 15 apresenta um exemplo desses procedimento de recorte de uma área da caixa de macha, buscando características únicas da peça.

Figura 15 – Segmentação via Labeling (versão 1.8.6)



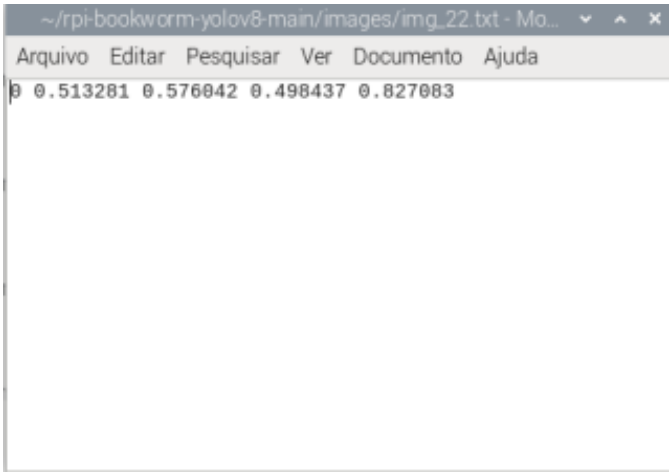
Fonte: Elaborado pelo autor

Para a anotação e preparação do conjunto de dados, foi utilizado o software Labeling (versão 1.8.6), uma ferramenta gráfica que permite marcar manualmente objetos de interesse em imagens. Esse recurso foi essencial para seccionar as imagens capturadas, definindo com precisão as áreas que continham peças industriais, como caixas e suportes. Além da delimitação das regiões, o Labeling também possibilitou a seleção de pontos importantes nas imagens, gerando automaticamente os pontos de coordenadas dos pixels para cada área anotada. Isso permitiu a identificação exata das posições dos objetos dentro das imagens. As anotações foram exportadas em formato compatível com o modelo YOLOv8, garantindo uma integração eficiente entre a etapa de rotulagem e o treinamento da rede neural

Durante o processo de rotulagem, foram gerados arquivos no formato **.**txt**,** nos quais são armazenadas as coordenadas das caixas delimitadoras (*bounding boxes*) para cada objeto identificado nas imagens. Esses arquivos contêm informações precisas

sobre a posição dos objetos. A figura 16 mostra este arquivo com as coordenadas do recorte da área de interesse.

Figura 16 – Dados de coleta da segmentação via Labeling



Fonte: Elaborado pelo autor

O uso dessa ferramenta foi essencial para garantir a qualidade das anotações, pois possibilitou a marcação visual direta sobre a imagem, contribuindo para a construção de um conjunto de dados bem estruturado e compatível com o modelo YOLOv8. A fidelidade visual entre as peças reais e suas marcações reforça a confiabilidade do processo de rotulagem manual.

3.3 Treinamento

Foi gerado um arquivo compactado no formato **.**zip contendo todas as imagens e seus respectivos arquivos de anotação, organizados conforme o padrão exigido pelo YOLO. Esse arquivo facilitou a aplicação do conjunto de dados no treinamento do modelo, garantindo a correta leitura e associação entre as imagens e as informações de rotulagem. YOLOv8 foi treinado utilizando um conjunto de dados dividido em três partes: treinamento, validação e teste.

Essa divisão visou garantir uma avaliação imparcial e realista da capacidade do modelo em identificar e classificar corretamente as peças. Durante o treinamento, o

modelo ajustou seus parâmetros com base nas imagens rotuladas, aprendendo a reconhecer os padrões visuais das classes envolvidas.

Inicialmente, o treinamento foi realizado com 100 imagens, sendo posteriormente executado com variações na quantidade de imagens, nas tipologias de peças selecionadas e também no número de épocas. Essa abordagem permitiu analisar como diferentes configurações de dados e de parâmetros influenciam no desempenho final do modelo.

Durante o treinamento, o modelo processa cada imagem, identifica as características visuais relevantes e ajusta seus pesos com base nos erros calculados a partir das diferenças entre as bounding boxes previstas e as verdadeiras, além das classificações das peças. Esse ajuste é realizado através de algoritmos de otimização, como o gradiente descendente, que iterativamente aprimoram a capacidade do modelo de detectar e classificar corretamente os objetos.

3.4 Validação

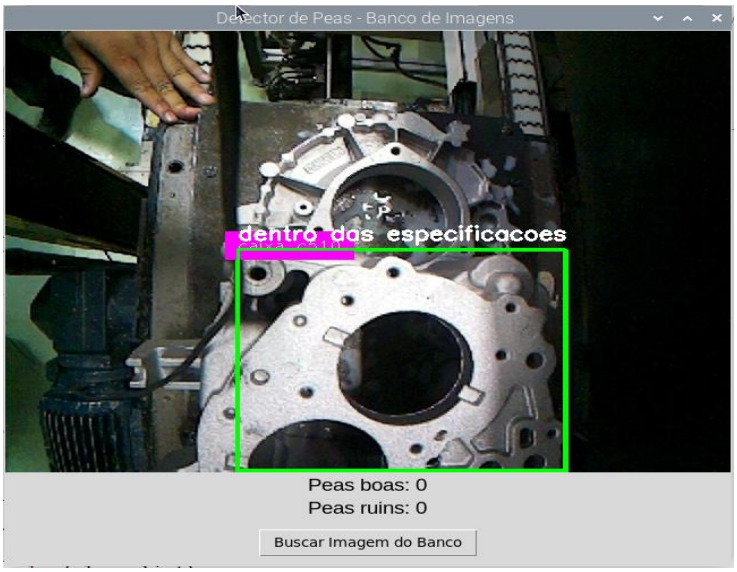
O processo de treinamento, o modelo YOLOv8 gera um arquivo denominado best.pt, que contém os pesos do modelo com o melhor desempenho obtido até aquele ponto. Esse arquivo representa a versão otimizada do modelo, capaz de realizar detecções e classificações com maior precisão com base nos dados de treinamento e validação.

O arquivo best.pt é fundamental para a aplicação prática do sistema, pois é utilizado na etapa de inferência para identificar as peças industriais em imagens novas, garantindo que o modelo empregue os parâmetros mais eficazes aprendidos durante o treinamento.

Importante destacar que a validação foi realizada em ambiente controlado, utilizando imagens previamente selecionadas que simulavam diferentes condições de iluminação, posicionamento e enquadramento. Essa abordagem teve como objetivo testar a robustez do modelo diante de variações comuns no processo industrial, mas sem interferências externas ou em tempo real.

Foi desenvolvido um algoritmo, integrado a uma interface homem-máquina (IHM), para realizar a validação do modelo utilizando as 100 imagens restantes do banco de dados. Esse sistema permitiu avaliar o desempenho do modelo em condições controladas, facilitando a análise dos resultados e a identificação de possíveis melhorias. Como pode ser visto na Figura 17, a interface exibe o status das peças, indicando se estão em conformidade ou não. Essa visualização facilita o acompanhamento dos resultados e contribui para o cálculo eficiente das métricas de desempenho do modelo.

Figura 17 – IHM do sistema de visão



Fonte: Elaborado pelo autor

As métricas utilizadas para avaliar o desempenho do modelo foram precisão (precision), revocação (recall), F1-score e mAP (mean Average Precision), todas geradas automaticamente pela ferramenta Ultralytics YOLOv8. Esses indicadores permitiram verificar a qualidade das detecções e a consistência das classificações realizadas.

Para complementar a validação, as previsões geradas pelo modelo foram comparadas com as anotações originais realizadas no software Labeling. Essa comparação teve como foco verificar se as bounding boxes previstas coincidiam com as regiões efetivamente rotuladas, garantindo a confiabilidade espacial do sistema e confirmando sua aplicabilidade futura em ambientes reais.

4 RESUTADOS E DISCUSSÕES

Na **primeira fase do teste**, foi utilizado um modelo treinado com um banco de dados contendo 100 imagens e submetido a 50 épocas de treinamento.

Foram realizados três testes utilizando o banco de imagens desenvolvido para o projeto.
O primeiro teste foi conduzido com 100 imagens, contemplando três situações distintas, com a seguinte distribuição:

- Peça conforme: 60 exemplos, 59 falsos positivos, 1 falso negativo.
- Peça ruim (não conforme): 15 exemplos, 9 Falsos positivos, 4 verdadeiros negativos.
- Ausência de peça (sem objeto na área de inspeção): 35 exemplos, 30 falsos positivo, 5 Verdadeiros negativos.

Na **segunda fase do teste**, foi utilizado um modelo treinado com um banco de dados contendo 80 imagens e submetido a 100 épocas de treinamento.

Foram realizados três testes utilizando o banco de imagens desenvolvido para o projeto.
O primeiro teste foi conduzido com 100 imagens, contemplando três situações distintas, com a seguinte distribuição:

- Peça conforme: 60 exemplos, 55 falsos positivos, 5 falso negativo.
- Peça ruim (não conforme): 15 exemplos, 5 Falsos positivos, 10 verdadeiros negativos.
- Ausência de peça (sem objeto na área de inspeção): 35 exemplos, 10 falsos positivo, 25 Verdadeiros negativos.

Na **terceira fase do teste**, foi utilizado um modelo treinado com um banco de dados contendo 60 imagens e submetido a 100 épocas de treinamento.

Foram realizados três testes utilizando o banco de imagens desenvolvido para o projeto.
O primeiro teste foi conduzido com 100 imagens, contemplando três situações distintas, com a seguinte distribuição:

- Peça conforme: 60 exemplos, 40 falsos positivos, 20 falso negativo.
- Peça ruim (não conforme): 15 exemplos, 10 Falsos positivos, 15 verdadeiros negativos.
- Ausência de peça (sem objeto na área de inspeção): 35 exemplos, 20 falsos positivo, 15 Verdadeiros negativos.

Após as três fases de teste em banco de imagens e implementação do sistema de visão computacional com o modelo YOLOv8 integrado ao Raspberry Pi, foram conduzidos testes em ambiente real de produção, com o objetivo de avaliar a eficácia da solução proposta utilizando o treinamento da fase 2. A Figura 18 apresenta a configuração do hardware montado acima da linha de inspeção, posicionada estrategicamente para capturar imagens das peças em movimento.

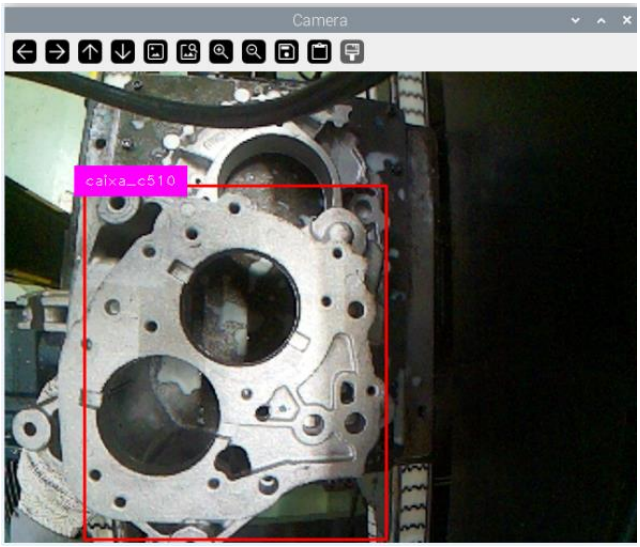
Figura 18 – Sistema em funcionamento em ambiente industrial

Fonte: Elaborado pelo autor

Foi realizado um teste de campo em ambiente industrial. Mesmo diante de desafios como variações de iluminação, poeira, vibrações mecânicas e mudanças

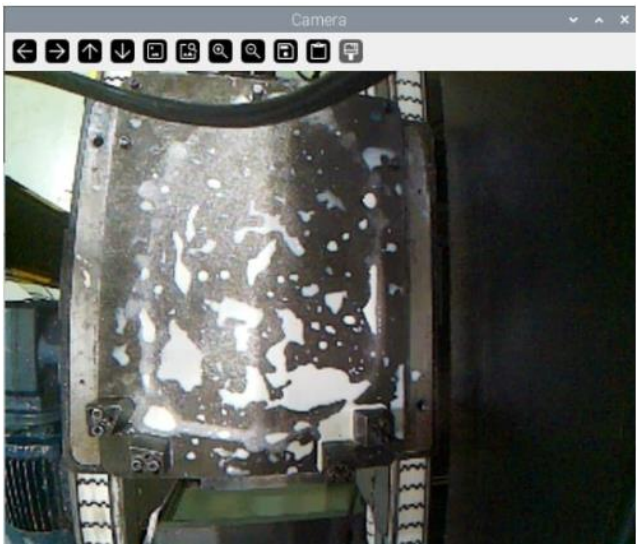
rápidas no posicionamento das peças, o sistema manteve desempenho estável. A Figura 18 ilustra um exemplo de detecção correta de uma peça sem falhas, enquanto a Figura 19 mostra a linha de produção sem peças, cenário no qual o sistema também respondeu corretamente ao não realizar nenhuma detecção.

Figura 16 – Teste do sistema do sistema de visão sem falhas.



Fonte: Elaborado pelo autor

Figura 17 – Teste do sistema



Fonte: Elaborado pelo autor

4.1 Desempenho dos três modelos Modelo

Fase	Épocas	Desempenho Geral	Pontos Fortes	Pontos Fortes
1ª Fase	50	Alta sensibilidade (baixo FN), mas muitos falsos positivos (FP)	Detecta bem as peças conformes (FN baixo)	Muitos falsos positivos, baixa precisão
2ª Fase	100	Melhor equilíbrio entre precisão e recall	Redução significativa dos falsos positivos; melhor reconhecimento das outras classes	Leve aumento de falsos negativos
3ª Fase	100	Sensibilidade reduzida (alto FN), falsos positivos diminuídos	Menos falsos positivos em peça conforme	Aumento dos falsos negativos e desempenho geral inferior devido a menos dados

- A segunda fase apresenta o melhor desempenho geral, com um bom equilíbrio entre falsos positivos e falsos negativos, além de melhor reconhecimento das classes “peça ruim” e “ausência de peça”.
- A primeira fase, embora tenha mais dados, teve poucas épocas de treinamento, resultando em muitos falsos positivos.
- A terceira fase teve mais épocas, mas menos dados, o que prejudicou a capacidade de generalização e resultou em maior número de falsos negativos.

4.2 Desempenho em Tempo Real

A média de tempo de inferência para cada imagem foi de 0,25 segundos, o que permitiu a operação em tempo real sem interrupções no fluxo de inspeção da linha de produção.

5 CONCLUSÃO

O objetivo deste trabalho foi o desenvolvimento de um sistema computacional para auxiliar na seleção automática de peças, utilizando técnicas de visão computacional e o algoritmo YOLO como núcleo da detecção e classificação. A proposta buscou automatizar o processo de inspeção, reduzindo a dependência de métodos manuais e aumentando a eficiência e a precisão na identificação de peças conformes, não conformes e na ausência de peças na área de inspeção.

O projeto contemplou desde a coleta e rotulagem (labeling) das imagens até o recorte e preparação do banco de dados para o treinamento do modelo. O processo de labeling foi fundamental para garantir que o YOLO identificasse com precisão as regiões de interesse nas imagens, enquanto o recorte e a organização dos dados asseguraram um treinamento mais consistente e representativo.

Os testes foram realizados em três fases, variando o tamanho do conjunto de treinamento e o número de épocas, permitindo avaliar o impacto desses fatores no desempenho do modelo:

Primeira fase – Treinamento com 100 imagens e 50 épocas: apresentou resultados iniciais, mas com alta taxa de erros, principalmente falsos positivos.

Segunda fase – Treinamento com 80 imagens e 100 épocas: obteve o melhor equilíbrio entre precision e recall, com menor número de erros de classificação, demonstrando que o ajuste das épocas aliado a um dataset limpo e representativo potencializa o desempenho do modelo.

Terceira fase – Treinamento com 60 imagens e 100 épocas: desempenho inferior à segunda fase, com aumento nos erros, confirmando a importância de um conjunto de treinamento robusto e variado.

Dessa forma, conclui-se que o uso do YOLO no processo de inspeção automática de peças é viável e tecnicamente eficaz, desde que sustentado por um banco de dados suficientemente amplo e representativo e por um processo de ajuste fino nos parâmetros de treinamento. Para aplicações industriais, recomenda-se a ampliação da base de imagens e a diversificação dos cenários de teste, visando reduzir erros e melhorar a confiabilidade do sistema.

5.1 Trabalhos Futuros

Para trabalhos futuros, diversas melhorias podem ser implementadas para aumentar a eficiência e a usabilidade do sistema de visão computacional desenvolvido neste estudo. Primeiramente, é fundamental melhorar a interface de utilização do sistema, tornando-a mais intuitiva para o usuário.

Outra proposta é a integração direta do sistema ao processo produtivo, possibilitando inspeção em tempo real e geração de ações automáticas, como a parada eminente da linha ou alertas imediatos à equipe responsável. Além disso, a inclusão de um módulo para apresentação detalhada dos defeitos identificados em cada peça pode auxiliar na análise de falhas e na melhoria contínua do processo, evidenciando quais não conformidades foram detectadas.

Também é recomendada a implementação de armazenamento de dados em nuvem, viabilizando acesso remoto aos resultados, imagens e métricas, bem como a execução do sistema em ambiente de nuvem para maior poder computacional, escalabilidade e facilidade de manutenção. Por fim, a expansão do banco de dados com diferentes tipologias de peças contribuirá para a adaptação do modelo a novos cenários industriais, ampliando seu potencial de aplicação.

6 REFERÊNCIAS

AGHION, P.; HOWITT, P. **A model of growth through creative destruction.** Econometrica, v. 60, n. 2, p. 323–351, 1992.

ALBUQUERQUE, Siderley Fernandes et al. **Avaliação da microestrutura e propriedades mecânicas de metais de solda obtidos por processos de soldagem manual e automatizado utilizado na soldagem de aço API 5L X80.** Soldagem & Inspeção, v. 16, p. 322-332, 2011.

BRADSKI, G. **The OpenCV Library.** Dr. Dobb's Journal of Software Tools, 2000.

COELHO, Fagner Guilherme Ferreira. **Desenvolvimento de um sistema de visão de baixo custo utilizando um manipulador robótico industrial visando a automatização de processo de soldagem.** 2016.

CROSTA, Alvaro Penteado. **Processamento digital de imagens de sensoriamento remoto.** UNICAMP/Instituto de Geociências, 1999.

DENIS, CLÁUDIO; ASSIS, WÂNDERSON O. **Seleção de Tomates para Processamento Industrial por Meio de Redes Neurais Aplicadas em Sistema de Visão Computacional.** In: Anais do VIII Congresso Brasileiro de Redes Neurais. 2007.

GONZALEZ, R. C.; WOODS, R. E.; EDDINS, S. **Digital Image Processing Using MATLAB.** New Jersey: Prentice Hall, 609 p., 2004.

JOCHER, Glenn et al. **ultralytics/yolov5: v6.1-tensorrt, tensorflow edge tpu and openvino export and inference.** Zenodo, 2022.

LENZ, R.; TSAI, R. Y. **A New Technique for Fully Autonomous and Efficient 3D Robotics hand/Eye Calibration.** IEEE Transactions on Robotics and Automation, U.S.A., v. 5, n. 3, p. 345-358, Junho, 1989.

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

LIU, Guangrui. **Real-Time Object Detection for Autonomous Driving Based on Deep Learning.** Dissertação (Mestrado) – Texas A&M University-Corpus Christi, 2017.

LUDERMIR, T. B. **Inteligência Artificial e Aprendizado de Máquina: estado atual e tendências.** Estudos Avançados, SciELO Brasil, Pernambuco, v. 35, p. 85–94, 2021.

MORÉ, Jesús Domech. **Aplicação da lógica Fuzzy na avaliação da confiabilidade humana nos ensaios não destrutivos por ultra-som.** Tese (Doutorado) – Universidade Federal do Rio de Janeiro, COPPE, 2004.

OLIVO, Regis Yuri; PEDRO LUIZ DE PAULA FILHO; ARNALDO CANDIDO JUNIOR. **Uma abordagem neural na identificação de objetos em imagens para auxílio na manutenção de rede elétrica.** In: Anais Estendidos do XXXIII Conference on Graphics, Patterns and Images. SBC, 2020.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. **You Only Look Once: Unified, Real-Time Object Detection.** Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p. 779–788, 2016. DOI: 10.1109/CVPR.2016.91.

ROSENBLATT, F.; PAPERT, S. **Perceptron.** Vol. 9. April, 2021.

SILVA, Fábio Silveira et al. **Desenvolvimento de um controle inteligente para seleção de mangas aplicada a um protótipo de manufatura robotizada.** Mostra Nacional de Robótica, Curitiba, 2017.

SIEGWART, R.; NOURBAKHSH, I. **Introduction to Autonomous Mobile Robots.** MIT Press, Cambridge, 2004. Capítulo 4.

SONKA, M.; HLAVAC, V.; BOYLE, R. **Image Processing, Analysis, and Machine Vision.** 4th ed. Boston: Cengage Learning, 2014.

VASCONCELOS, L. M. et al. **FRAMEWORK PARA CONVERTER IMAGENS RGB PADRÃO EM DADOS NEUROMÓRFICOS.** movimento, v. 16, p. 17.

VENKATKUMAR, Y. **YoloV8 Architecture & Cow Counter With Region Based Dragging Using YoloV8.** 2023. Disponível em: . Acesso em: 01 dez 2023.

ZHANG, A. et al. **Dive into Deep Learning.** Cambridge University Press, 2023.

APENASIMAGENS. **Pixel imagem digital.** Disponível em: https://apenasimagens.com/pt/pixel-imagem-digital/

7 APÊNDICE A:

Código de formação do banco de dados

```
from picamera2 import Picamera2
import time

picam2 = Picamera2()
picam2.preview_configuration.main.size = (640,480)
picam2.preview_configuration.main.format = "RGB888"
picam2.preview_configuration.align()
picam2.configure("preview")
picam2.start()
cpt = 0
maxFrames =150
while cpt < maxFrames:
    im= picam2.capture_array()
    im=cv2.flip(im,-1)
    cv2.imshow("Camera", im)
    cv2.imwrite('/home/rael/rpi-bookworm-yolov8-main/images/img_%d.jpg' %cpt, im)
    cpt += 1
    if cv2.waitKey(1)==ord('q'):
        break
cv2.destroyAllWindows()
```

Apendice B código do sistema de visão

```
from ultralytics import YOLO
import cv2

# Carrega o modelo treinado
model = YOLO("best.pt")

# Acessa a webcam (ou câmera USB / CSI)
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    results = model(frame, stream=True)

    for r in results:
        annotated_frame = r.plot()

    cv2.imshow("Detecção de Peças", annotated_frame)

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

cap.release()
cv2.destroyAllWindows()
```