UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Tecnologia

Rodrigo Braz Rodrigues

O USO DE REDES NEURAIS ARTIFICIAIS NA ANÁLISE ÓPTICA AUTOMÁTICA E DETECÇÃO DE DEFEITOS EM CIRCUITOS IMPRESSOS

Limeira
2023

Rodrigo Braz Rodrigues

O USO DE REDES NEURAIS ARTIFICIAIS NA ANÁLISE ÓPTICA AUTOMÁTICA E DETECÇÃO DE DEFEITOS EM CIRCUITOS IMPRESSOS

Dissertação apresentada à Faculdade de Tecnologia da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Tecnologia, na área de Sistemas de Informação e Comunicação.

Orientador: Prof. Dr. Varese Salvador Timoteo

Este trabalho corresponde à versão final da Dissertação defendida por Rodrigo Braz Rodrigues e orientada pelo Prof. Dr. Varese Salvador Timoteo.

Limeira
2023

FOLHA DE APROVAÇÃO

Abaixo se apresentam os membros da comissão julgadora da sessão pública de defesa de dissertação para o Título de Mestre em Tecnologia na área de concentração Sistemas de Informação e Comunicação, a que se submeteu o aluno Rodrigo Braz Rodrigues, em 16 de novembro de 2023 na Faculdade de Tecnologia – FT/UNICAMP, em Limeira/SP.

Prof. Dr. Varese Salvador Timoteo
Presidente da Comissão Julgadora

Profa. Dra. Luciana Claudia de Paula
Universidade Estadual De Santa Cruz/UESC

Dr. Rafael Fernando Diório
Instituto Federal de São Paulo/IFSP

Ata da defesa, assinada pelos membros da Comissão Examinadora, encontra-se no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós-graduação da Faculdade de Tecnologia.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Faculdade de Tecnologia
Felipe de Souza Bueno - CRB 8/8577

Rodrigues, Rodrigo Braz, 1988-
R618u O uso de redes neurais artificiais na análise óptica automática e detecção de defeitos em circuitos impressos / Rodrigo Braz Rodrigues. – Limeira, SP : [s.n.], 2023.

Orientador: Varese Salvador Timóteo.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Tecnologia.

1. Controle de qualidade - Métodos óticos. 2. Placas de circuito impresso. 3. Redes neurais (Computação). 4. Conjunto de dados. I. Timóteo, Varese Salvador, 1972-. II. Universidade Estadual de Campinas. Faculdade de Tecnologia. III. Título.

Informações Complementares

Título em outro idioma: The use of artificial neural networks in analysis automatic optics and defect detection in printed circuits boards
Palavras-chave em inglês:
Quality control - Optical methods
Printed circuit boards
Neural networks (Computer science)
Data sets
Área de concentração: Sistemas de Informação e Comunicação
Titulação: Mestre em Tecnologia
Banca examinadora:
Varese Salvador Timóteo [Orientador]
Luciana Claudia de Paula
Rafael Fernando Diório
Data de defesa: 16-11-2023
Programa de Pós-Graduação: Tecnologia

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0009-0007-0785-9351
- Currículo Lattes do autor: http://lattes.cnpq.br/0206947939978128

Não podemos resolver um problema usando o mesmo tipo de pensamento que usamos quando os criamos.

(Albert Einstein)

Agradecimentos

Agradeço à minha amada esposa Alessandra, cujo amor, paciência e apoio inabaláveis sustentou minha dedicação a este projeto. Sua compreensão, incentivo constante e apoio emocional foram essenciais para superar os desafios e dificuldades que enfrentei durante essa jornada. Ao meu filho Raphael, minha força motriz. Ao meu respeitado orientador, Dr Varese, minha mais sincera gratidão. Sua orientação experiente, visão acadêmica e dedicação ao meu crescimento como pesquisador foram cruciais para o desenvolvimento deste trabalho. Além disso, gostaria de agradecer a todos os amigos, cujas discussões, insights e amizade fizeram desta jornada uma experiência enriquecedora e memorável. Minha gratidão à Faculdade de Tecnologia da Universidade Estadual de Campinas pela oportunidade. A todos vocês, meu sincero obrigado. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Resumo

A presente dissertação contempla uma proposta de *software* gerador de conjunto de dados, *datasets*, para a detecção de defeitos em placas de circuito impresso, PCIs, por meio da criação de *datasets* sintéticos. A detecção de defeitos em PCIs é uma tarefa crítica na produção de equipamentos eletroeletrônicos, exigindo métodos robustos e eficientes de Inspeção Óptica Automatizada. No entanto, a disponibilidade de conjuntos de dados reais suficientemente diversificados muitas vezes se torna um desafio. Portanto esta pesquisa propõe a geração de *datasets* sintéticos como uma solução para superar as limitações dos *datasets* públicos disponíveis. Utilizando técnicas de sobreposição de imagens e combinação simples, para criação de amostras representativas de defeitos com base em uma variedade de defeitos de PCIs presentes na literatura. A abordagem sintética permite criar e ampliar significativamente a diversidade e a complexidade dos datasets, proporcionando uma melhor generalização dos modelos de detecção treinados. A avaliação da proposta é realizada por meio de prova de conceito aplicando a detecção de defeitos presentes no *dataset* sintético, somados a anotações padronizadas utilizados para treinamento. Por fim, espera-se que a proposta de *software* contribua como uma nova ferramenta para criação de *datasets* sintéticos voltados a inspeção óptica automática de defeitos em PCIs.

Palavras chave: AOI, *Dataset* sintéticos, Defeitos de Placas de Circuito Impresso

Abstract

The present dissertation contains a proposal for dataset generation software, focusing on datasets for detecting defects in *Printed Circuit Boards* (PCBs) by creating synthetic datasets. Detecting defects in PCBs is critical in producing electronic equipment, requiring robust and efficient methods of Automated Optical Inspection (AOI). However, the availability of sufficiently diverse real datasets often becomes a challenge. Therefore, this research proposes the generation of synthetic datasets as a solution to overcome the limitations of publicly available datasets. Using image overlay and simple combination techniques, it aims to create representative samples of defects based on various PCB defects found in the literature. The synthetic approach allows for the creation and significant expansion of dataset diversity and complexity, providing better generalization for trained defect detection models. The proposal is evaluated through a proof of concept by applying defect detection on the synthetic dataset, combined with standardized annotations used for training. In conclusion, The proposed software tool can create customized synthetic datasets for identifying defects in PCBs.

Keywords: AOI, Synthetic dataset, Printed Circuit Board Defects

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AOI	*Automated Optical Inspection*
CNN	*Convolutional Neural Networks*
CPU	*Central Processing Unit*
GANs	*Generative Adversarial Networks*
GPU	*Graphics Processing Unit*
IA	Inteligência Artificial
IDE	*Integrated Development Environment*
mAP	*Mean Average Precision*
ML	*Machine Learn*
MVC	*Model View Control*
P	*Precision*
PCBAs	*Printed Circuit Boards Assembly*
PCBs	*Printed Circuit Boards*
PCIs	Placas de Circuito Impresso
R	*Recall*
RNA	Rede Neural Artificial
SMD	*Surface Mounted Device*
SMT	*Surface-Mount Technology*
SPI	*Solder Paste Inspection*
YOLO	*You Only Look Once*

Sumário

Capítulo 1

Introdução

Nos últimos anos, a Inteligência Artificial está cada vez mais presente no nosso dia a dia, seja em aplicações de *smartphones*, computadores, efeitos em vídeos e fotos, ChatGPT, recomendação de filmes em plataformas de *streaming* etc. Para que tudo isso seja possível, estas aplicações contam com sistemas de Inteligência artificial, dentre elas as redes neurais artificiais, que são sistemas complexos capazes de identificar padrões, realizar predições etc.

Na indústria, é necessário um sistema que consiga garantir a qualidade, por meio de análise, de um grande volume de produtos, em virtude da rápida atividade produtiva, e nisto o uso de inspeção óptica automatizada, *Automated Optical Inspection* (AOI) , oferece vantagens ao comparar com análises humanas.

A indústria de eletroeletrônicos utiliza em seus produtos Placas de Circuito Impresso (PCIs), responsáveis por grande parte das funcionalidades de seus produtos. Durante a fabricação de PCIs, defeitos podem ser gerados e a detecção deles garantem a melhor qualidade dos produtos e a redução de processamento adicional em PCIs defeituosas. Para detecção de defeitos nestas placas, a utilização de redes neurais artificiais vem sendo estudada conforme publicado nos trabalhos, (WANG, C. et al., 2023a; SCHWEBIG, A.; TUTSCH, 2020)

Para que haja sucesso no aprendizado de máquina e consequentemente bons resultados na detecção e classificação de defeitos são necessários conjunto de dados, *Dataset*, que representem adequadamente os defeitos das PCIs (WANG, C. et al., 2023b).

Neste contexto, este projeto de pesquisa propõe o desenvolvimento de um *software* capaz de gerar *datasets* sintéticos aptos a serem utilizados para atividade de aprendizado de máquina para aplicações de AOI.

Diferentes de outros projetos, onde os *datasets* são aumentados utilizando Aumento de Dados, *Data Augmentation*, ou Redes Generativas Adversariais, *Generative Adversarial Networks* (GANs), ou simples geradores de anotações. Este trabalho propõe a criação de um *software* para geração de *datasets*, imagens de PCIs e suas respectivas anotações, a partir de combinações simples entre imagens dos componentes das PCIs e suas variações com e sem defeitos sobrepostos a imagem de uma PCI.

Para este propósito, quanto a organização desta dissertação, os conceitos básicos relacionados a contextualização do tema estão contidos no Capítulos 2, bem como a proposta de *software*, Capítulo 3 - Seção 3.2, a criação de um modelo de aprendizado de máquina no Capítulo 3 - Seção 3.4 seguidos da prova de conceito, Capítulo 3 - Seção 3.5, e por fim uma conclusão obtida após os testes de detecção, Capítulo 4.

Desta forma, espera-se, que a proposta de *software* contida neste trabalho contribua como uma nova ferramenta para criação de dados sintéticos para utilização em modelos de AOI.

1.1 Motivação

A produção de Equipamentos Eletroeletrônicos (EEE), tem aumentado de forma acelerada. Novos eletroeletrônicos são lançados a todo momento para as mais variadas finalidades e formatos, desde um tag de geolocalização, *smartphones*, computadores, carros etc.

EEE possuem PCIs, responsáveis pelos percursos das correntes, sustentação de seus componentes e lógica embarcada.

Em escala industrial, estas PCIs são produzidas com a tecnologia *Surface-mount technology*, SMT, capaz de promover a rápida fabricação de PCIs complexas, densas e com componentes pequenos, o que torna a inspeção visual humana uma tarefa difícil. Por esta razão sistemas de AOI são de suma importância para detecção de defeitos, por possibilitar a eliminação de causas de defeitos, análise da produção, redução de custos e a diminuição de retrabalhos. A detecção de defeitos na manufatura é extremamente importante para garantir a qualidade dos produtos fabricados. A detecção precoce de defeitos pode ajudar a reduzir os custos de produção, pois evita que os produtos defeituosos sejam enviados para próximas estações de montagem de EEE, e posteriormente ao mercado.

A SMT utiliza a AOI como solução de automação para tarefas de detecção e classificação dos defeitos ocasionados no processo de fabricação de PCI. Para realização destas tarefas os

sistemas de AOI que utilizam algoritmos de aprendizado de máquina requerem um conjunto de dados, *Dataset*, com imagens rotuladas utilizadas para ensinar o sistema a reconhecer padrões relacionados aos defeitos.

Durante as pesquisas deste trabalho foram encontrados *Datasets* com PCIs em três condições:

- PCIs sem componentes, impossibilitando a classificação de defeitos relacionados aos mesmos (AKHATOVA, 2021; HUANG; WEI, 2019);
- PCIs com componentes, porém com PCIs destinadas a reciclagem, cujo funcionamento e estado de conservação estão diferentes do obtido em uma linha produção (PRAMERDORFER; KAMPEL, 2015);
- *Datasets* sob sigilo, cujo uso é restrito a um estudo específico (SCHWEBIG, A.; TUTSCH, 2020);

Além destas condições, trabalhos como o de (SCHWEBIG, A.; TUTSCH, 2020; WANG, C. et al., 2023b) e (YU; HE, Y., 2022) confirmam escassez de *datasets* e o quanto a aprendizagem profunda é significativamente afetada devida a falta de amostras para aplicações de AOI.

Como solução a escassez de *datasets* este trabalho propõe o desenvolvimento de um *software* capaz de gerar *dataset* sintético para aplicações de AOI, composto por imagens de PCIs e anotações no formato YOLO Darknet TXT.

1.2 Objetivo

Esta dissertação surge da crescente necessidade de adaptação da indústria de EEE quanto a garantia da qualidade de seus componentes PCIs diante de inúmeras possibilidades de construção. Aprofundar a compreensão e abordagem em relação a mecanismos de automação de sistemas de visão computacional para inspeção visual é uma necessidade. Diante do contexto atual de baixa disponibilidade de *datasets* públicos, relacionados a PCIs e seus defeitos, e a alta importância de sua existência para aplicações de AOI os objetivos são:

1.2.1 Objetivo Geral

Propor o desenvolvimento de um *software* capaz de gerar *dataset* de PCIs a partir da imagem de uma Placas de Circuito Impresso Nua, *Printed Circuit Board*, PCB, sem componentes e que permita a inclusão de componentes com e sem defeitos sobrepostos a imagem da PCB formando Placas de Circuito Impresso Funcionais, *Printed Circuit Boards Assembly* (PCBAs), novas conforme a sua composição. Os detalhes de implementação do objetivo geral, objetivos específicos e artefatos de entrada e saída estão no Capítulo 3.

1.2.2 Objetivos Específicos

1. Criar casos de usos para compreensão da proposta de *software* para demonstrar em alto nível o que é esperado, quais são as funcionalidades necessárias e como os atores do sistema interagem;

2. Criar o diagrama de classes juntamente com a arquitetura do *software* proposto;

3. Criar requisitos funcionais da proposta de *software*;

4. Treinar e utilizar um modelo de rede neural convolucional, YOLOv5, análogo a um existente na AOI para exemplificar a possibilidade do uso do *dataset* como demonstração de detecção;

5. Demonstrar e correlacionar as anotações do *dataset* de saída do *software* proposto com o *dataset* público utilizado para o treinamento da rede mencionada no Objetivo Específico 4.

Capítulo 2

Levantamento bibliográfico

2.1 Inteligência Artificial

Embora Inteligência Artificial (IA) seja um tema em alta na atualidade o conceito de usar computadores na simulação de comportamento inteligente foi descrito pela primeira vez por Alan Turing em 1950, no livro *Computing machinery and intelligence*, Turing descreveu um teste simples, que mais tarde ficou conhecido como "Teste de Turing"(KAUL; ENSLIN; GROSS, 2020) cujo objetivo é determinar se os computadores eram capazes de gerar inteligência humana, a proposta de seu teste consiste em avaliar a inteligência de uma aplicação, onde um interrogador realiza perguntas e obtém respostas cuja origem, humana ou máquina, não pode ser distinguida.(RUSSELL; NORVIG, 2011)

Seis anos depois, no verão de 1956 no Dartmouth College, Hanover, New Hampshire, John McCarthy descreveu o termo inteligência artificial como *"a ciência e a engenharia de fazer máquinas inteligentes"* em um workshop destinado ao estudo da teoria de autômatos, redes neurais e estudo da inteligência (KAUL; ENSLIN; GROSS, 2020). Sua proposta era prosseguir com modelos de *Machine Learn* (ML), e mapear características da inteligência de tal maneira que pudesse reproduzir em uma máquina, tornando-a capaz de aperfeiçoar-se e resolver problemas que outrora somente os humanos seriam capazes de resolver (RUSSELL; NORVIG, 2011).

Em 1958 Frank Rosenblatt desenvolveu o *perceptron*, um modelo de neurônio artificial que podia aprender a classificar dados. Embora limitado a problemas linearmente separáveis, ele influenciou o desenvolvimento futuro de redes neurais.

O *perceptron* é um tipo de algoritmo de aprendizagem supervisionada para classificação binária (WESTNY, 2023; VU-QUOC; HUMER, 2023; ALIJAGIĆ; ŠAJN, 2020; TACCHINO et al., 2019; YU; SHU, 2021). Baseia-se no conceito de um único neurônio artificial ou *perceptron*, que pega um conjunto de características de entrada com pesos associados e os combina para produzir uma saída (WESTNY, 2023; VU-QUOC; HUMER, 2023; ALIJAGIĆ; ŠAJN, 2020; TACCHINO et al., 2019; YU; SHU, 2021).

O algoritmo *perceptron* de Rosenblatt foi inspirado na neurociência e tinha como objetivo imitar o comportamento dos neurônios biológicos (VU-QUOC; HUMER, 2023). Ele lançou as bases para o desenvolvimento de redes neurais artificiais, RNAs, *feed-forward* em camadas (ALIJAGIĆ; ŠAJN, 2020). O algoritmo *perceptron* é frequentemente referido como *perceptron* de camada única, pois consiste em uma única camada de neurônios artificiais (ALIJAGIĆ; ŠAJN, 2020).

O algoritmo *perceptron* é uma abordagem simples e intuitiva para tarefas de classificação binária. Ele ajusta iterativamente os pesos dos recursos de entrada com base nos erros cometidos na classificação, visando encontrar o limite de decisão ideal que separa as duas classes (WESTNY, 2023; VU-QUOC; HUMER, 2023; ALIJAGIĆ; ŠAJN, 2020; TACCHINO et al., 2019; YU; SHU, 2021). Porém, possui limitações, como a incapacidade de representar funções complexas como a função XOR (VU-QUOC; HUMER, 2023).

Com o tempo, o algoritmo *perceptron* foi estendido e aprimorado, levando ao desenvolvimento de arquiteturas de redes neurais e algoritmos de aprendizagem mais sofisticados (VU-QUOC; HUMER, 2023; ALIJAGIĆ; ŠAJN, 2020; TACCHINO et al., 2019; YU; SHU, 2021). No entanto, o *perceptron* continua sendo um marco importante na história do aprendizado de máquina e das redes neurais artificiais.

Anos depois, em 1964, Joseph Weizenbaum apresentou Eliza, que utilizava processamento de linguagem natural usando metodologia de correspondência e substituição de padrões para imitar uma conversa, dando suporte ao que hoje compreendemos como *chatterbots*. Em 1966, no Stanford Research Institute foi criado *"a primeira pessoa eletrônica"*, capaz de processar instruções complexas ao invés de comandos seriados (KAUL; ENSLIN; GROSS, 2020).

Entre as décadas de 1970 a 1980 houve 2 períodos chamados *"inverno da IA"*, nestes períodos o financiamento ao desenvolvimento de pesquisas foram reduzidos. Para o primeiro, logo após o final da década de 1970, a motivação foi a limitação percebida da IA frente ao entusiasmo gerado nos anos anteriores. Já o segundo ocorreu no final da década de

1980, devido ao custo excessivo no desenvolvimento e manutenção de bases de dados. Apesar da falta de interesse geral durante este período, estudos como o de Edward Feigenbaum impulsionaram o desenvolvimento da história da IA. Feigenbaum e sua equipe avançaram ainda mais no campo da IA com o desenvolvimento do sistema especialista MYCIN (KULIKOWSKI, 2020b). O MYCIN foi desenvolvido para auxiliar no diagnóstico e tratamento de doenças infecciosas, principalmente infecções bacterianas. Utilizou abordagens baseadas em regras para recomendar tratamentos apropriados com base nos sintomas do paciente e nos resultados dos testes laboratoriais (KAUL; ENSLIN; GROSS, 2020).

Durante a década de 1990, avanços notáveis da IA passaram a chamar a atenção do mundo como a vitória do computador *Deep Blue* sobre o campeão enxadrista Garry Kasparov e o protótipo do Google, capaz de indexar e classificar resultados.

Já a partir dos anos 2000, uma série de novas aplicações vêm sendo exploradas e passaram a fazer parte do dia a dia das pessoas em diversas aplicações, desde assistentes virtuais, como Alexa e a Siri, carros autônomos, como por exemplo o Tesla. (KULIKOWSKI, 2020a)

A Inteligência Artificial é um campo multidisciplinar que visa desenvolver máquinas inteligentes capazes de realizar tarefas que normalmente requerem inteligência humana (LEGG; HUTTER, 2007; LIU, N.; SHAPIRA; YUE, 2021). É uma tecnologia inovadora reconhecida mundialmente cujas aplicações estão presentes em vários setores, como saúde, educação, manufatura, cidades inteligentes, agricultura e muito mais. Na indústria de manufatura, por exemplo, a IA desempenha um papel vital nos processos de fabricação, melhorando a qualidade e reduzindo custos (RIZVI et al., 2021). Pode ser definida como um campo da engenharia da computação e ciência da computação com intuito de retratar computacionalmente processos de pensamento, raciocínio e comportamentos. Reproduzindo a forma como o ser humano pensa e age (RUSSELL; NORVIG, 2011). É um sistema inteligente capaz de realizar tarefas em um ambiente mutável e complexo, sem intervenção manual. É uma tecnologia que pode otimizar continuamente decisões e comportamentos por meio do aprendizado de máquina. Considerada um retrato da inteligência humana em computadores (HE, K. et al., 2016). Outra abordagem é definir IA como a capacidade de uma máquina perceber, raciocinar, aprender, comunicar e agir em ambientes complexos, de formas semelhantes ou possivelmente melhores que os humanos (LIU, N.; SHAPIRA; YUE, 2021; MONETT et al., 2020). Embora não exista uma definição universalmente aceita de IA, existe

uma convergência em atributos essenciais como a percepção, o raciocínio, a aprendizagem, a comunicação e a ação (LIU, N.; SHAPIRA; YUE, 2021).

Com base em (RUSSELL; NORVIG, 2011), dentre as disciplinas que compõe a IA, grande parte delas podem ser vistas a seguir:

- Processamento de linguagem natural: Ramo da IA que se concentra no desenvolvimento de sistemas capazes de entender a linguagem escrita ou falada e responder adequadamente a ela. Comumente estão relacionados a tarefas como tradução automática, compreensão do discurso e análise de sentimentos;
- Representação e armazenamento do conhecimento: Representação de informações em um formato, modelo computacional, e armazenamento destas informações de forma que elas possam ser acessadas, atualizadas, e utilizadas pelos algoritmos de IA;
- Aprendizado de máquina: Ramo da IA que visa o aprendizado a partir de dados. Os sistemas de aprendizado de máquina podem ser treinados, especializados, para o reconhecimento de padrões, previsões e classificações;
- Visão computacional, perceber e adaptar-se ao meio: Ramo da IA cuja concentração é a criação de sistemas capazes de interpretar imagens e vídeos. Atualmente é amplamente utilizada para detecção de objetos, reconhecimento facial, análise de movimento, monitoramento remoto etc;
- Raciocínio, dedução, indução, lógica;
- Robótica: Utilização de robôs capazes de desempenhar tarefas que de alguma forma envolvem as disciplinas descritas acima;

Concluindo, a IA é uma tecnologia que visa replicar a inteligência humana em computadores e permitir tomadas de decisão e comportamentos inteligentes em ambientes complexos. Tem diversas aplicações em vários campos, incluindo medicina, direito, educação, logística e design ambiental etc. A definição de IA pode variar entre diferentes disciplinas, mas geralmente refere-se ao desenvolvimento de sistemas inteligentes que podem imitar a inteligência humana.

2.2 Machine Learn

O *Machine Learn*, ML, é um subcampo da inteligência artificial que envolve o desenvolvimento de sistemas capazes de aprender a partir de dados (KÜHL et al., 2019). Utiliza algoritmos e modelos matemáticos para analisar e interpretar conjuntos de dados complexos, permitindo ao sistema fazer previsões ou decisões com base em padrões e tendências nos dados (LAMBA; INDIA; MISHRA, 2019; MARTINEZ; PEIRO-SIGNES, 2022). Um dos objetivos do aprendizado de máquina é criar modelos que possam melhorar automaticamente seu desempenho ao longo do tempo, sem serem explicitamente programados (KÜHL et al., 2019).

Existem dois tipos principais de aprendizado de máquina: aprendizado supervisionado e não supervisionado. A aprendizagem supervisionada envolve o treinamento de um modelo usando exemplos rotulados, onde os dados de entrada são emparelhados com a saída correspondente ou variável de destino. O modelo aprende com esses exemplos e pode então fazer previsões sobre dados novos e não vistos. Por outro lado, a aprendizagem não supervisionada envolve o treinamento de um modelo em dados não rotulados, onde o objetivo é descobrir padrões ou estruturas nos dados sem quaisquer rótulos predefinidos (KÜHL et al., 2019).

O processo de aprendizado de máquina normalmente envolve duas etapas: treinamento e teste. Durante a etapa de treinamento, o modelo é exposto a uma grande quantidade de dados rotulados e aprende a reconhecer padrões e relacionamentos entre as variáveis de entrada e saída. O modelo ajusta seus parâmetros ou pesos com base nos dados de treinamento, otimizando seu desempenho e minimizando erros. Uma vez treinado, o modelo pode ser testado em dados inéditos para avaliar seu desempenho e capacidade de generalização (MUSTIKA; NENDA; RAMADHAN, 2021).

Algoritmos de aprendizado de máquina podem ser aplicados a vários domínios. Esses algoritmos têm a capacidade de analisar conjuntos de dados complexos e fazer previsões ou decisões precisas além do que um especialista humano poderia fazer. Podem descobrir padrões e *insights* ocultos nos dados, permitindo que as organizações tomem decisões baseadas em dados e melhorem os seus processos (MARTINEZ; PEIRO-SIGNES, 2022).

Em resumo, o aprendizado de máquina é um ramo da inteligência artificial que se concentra no desenvolvimento de sistemas capazes de aprender a partir de dados. Envolve o uso de algoritmos e modelos matemáticos para analisar e interpretar conjuntos de dados complexos, permitindo ao sistema fazer previsões ou decisões com base em padrões e

tendências nos dados. O aprendizado de máquina pode ser categorizado em aprendizado supervisionado e não supervisionado e envolve um processo de treinamento e teste. Os algoritmos de aprendizado de máquina têm uma ampla gama de aplicações e podem fornecer *insights* e previsões valiosas em vários domínios (KÜHL et al., 2019).

2.3 Rede Neural Artificial

Uma Rede Neural Artificial (RNA), é um subcampo de ML, é um modelo computacional inspirado no funcionamento de redes neurais biológicas, como o cérebro humano, que é capaz de processar informações e realizar tarefas de aprendizado. A rede neural biológica é composta por conjuntos de neurônios, unidades básicas de processamento, interligados através de conexões sinápticas que ao receber estímulos do meio obtém informações, processam e constroem o aprendizado (HAYKIN, 2001)

As RNAs também são compostas por um conjunto de unidades de processamento interconectadas, chamadas de neurônios artificiais, que trabalham em conjunto para resolver problemas complexos. Assim como o cérebro humano, as RNAs são capazes de aprender a partir de exemplos e ajustar suas conexões internas para melhorar o desempenho em tarefas específicas. Elas são treinadas usando algoritmos de aprendizado, nos quais os pesos das conexões entre os neurônios são ajustados com base nos dados de entrada e nas saídas desejadas. Nas RNAs, os dados de entrada são recebidos em forma de arquivos, análogo aos dados obtidos por receptores biológicos, que ao serem processados e submetidos ao modelo computacional, permitem ao computador reproduzir tarefas como aprendizado, reconhecimento de padrões, classificações, predições, aproximações (HAYKIN, 2001).

Um dos precursores da RNA é o *Perceptron*, Figura 2.1, e a percepção de neurônio artificial, como uma unidade de processamento, pode ser entendida na proposta de Rosenblatt (ROSENBLATT, 1958).

Na prática, o neurônio artificial recebe os sinais de entrada e multiplica pelos pesos das conexões, realiza a junção somatória dos produtos da multiplicação anterior, executa a função de ativação, que por conseguinte gera uma saída binária que representa a ativação do neurônio.

Como o nome sugere, uma RNA é composta por uma série de neurônios artificiais interligados que processam a entrada e compartilham a saída com a próxima camada de

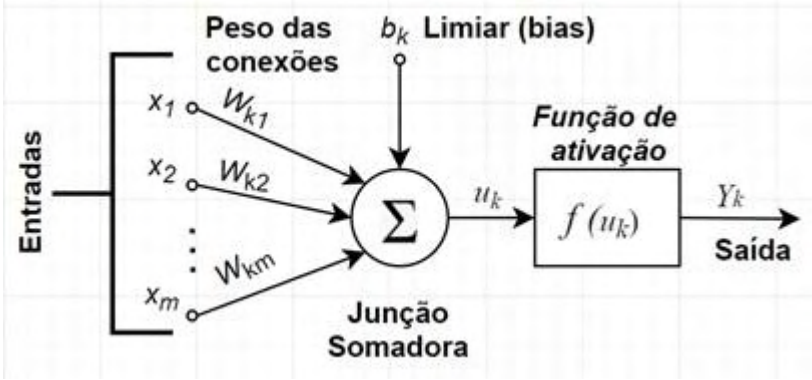


Figura 2.1: Modelo matemático de neurônio artificial com índice k - Adaptado de HAYKIN, 2001

neurônios, Figura 2.2, até que as informações cheguem à camada de saída e produzam o resultado.

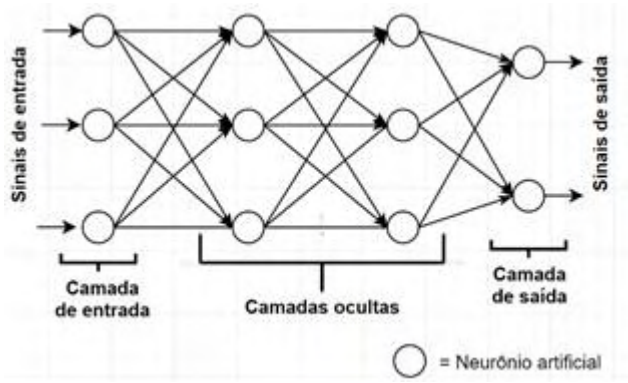


Figura 2.2: Representação de uma Rede Neural Artificial - Adaptado de HAYKIN, 2001

Uma outra característica das RNAs é que ao contrário dos programas de computador comuns, onde eles são desenvolvidos para realizar tarefas pré-definidas, as RNAs permitem que o computador aprenda a partir de dados observacionais e cumpra sua tarefa (NIELSEN, 2022).

Com relação às aplicações, as RNAs têm sido eficazes na resolução de problemas nos quais a solução não pode ser facilmente expressa por meio de regras ou algoritmos tradicionais. Uma das principais vantagens dos RNAs é sua capacidade de lidar com dados complexos e não lineares. Elas são capazes de aprender representações hierárquicas e não lineares dos dados, permitindo que capturem relações complexas entre as variáveis de entrada e saída. No entanto, as RNAs também apresentam algumas limitações. Elas podem exigir grandes quantidades de dados de treinamento para obter um bom desempenho e podem ser sensíveis a ruídos nos

dados. Além disso, o treinamento de RNAs pode ser computacionalmente intensivo e exigir recursos computacionais significativos.

2.3.1 Treinamento, testes e validação da RNA

O treinamento de uma RNA é o processo pelo qual o aprendizado é construído. O objetivo desta etapa é ajustar os pesos e parâmetros internos da rede para que ela seja capaz de realizar a tarefa ao qual está dedicada. Este processo visa a constante aproximação das previsões obtidas pelo treinamento em relação aos resultados reais, com o objetivo de tornar a RNA capaz de fazer previsões ou tomar decisões adequadas a dados ainda não vistos durante esta etapa.

O treinamento de uma RNA é feito com uma parcela do conjunto de dados, *dataset*, denominados dados de treinamento, estes dados devem representar a distribuição de valores similares a distribuição do *dataset* como um todo. Na sequência para cada um dos elementos do conjunto de treinamento são calculados os pesos e o bias com intuito de estimar um resultado próximo ao valor real. O cálculo dos pesos é feito de forma iterativa com valores iniciais aleatórios e são ajustados por meio de mecanismo de otimização. Ao terminar a etapa de treinamento um modelo é gerado.

Assim como o treinamento, o teste também contém uma parcela do *dataset*, porém esta parcela não foi apresentada ao treinamento. O objetivo do teste é a comprovação de que o modelo gerado pelo treinamento funciona. Já o conjunto de validação é utilizado para comparação entre modelos (YAMASHITA et al., 2018).

2.4 Rede Neural Convolucional

Redes neurais convulsionais, *Convolutional Neural Networks* (CNN), correspondem a um tipo de RNA de aprendizado profundo proposta inicialmente por Yann Lecun. Este tipo de RNA tem sido amplamente utilizada no processamento de informações visuais, imagens e vídeos, na tarefa de classificação, pois são especializadas em processar dados cuja topologia é semelhante a uma matriz, como é o caso da imagem que pode ser considerada como uma matriz 2D de pixels. O que caracteriza este tipo de RNA é o uso de camadas convolucionais, responsáveis pela aplicação de filtros que evidenciam características relevantes das imagens tornando possível sua classificação (LECUN et al., 1998; GOODFELLOW; BENGIO; COURVILLE, 2016; COSTA G. B. P.; PONTI, 2017).

A classificação, no escopo de visão computacional, consiste em identificar a qual classe pertence a imagem de entrada ou a probabilidade de pertencimento a determinada classe. Embora esta seja uma tarefa fácil para nós humanos, para o computador não é uma tarefa tão fácil, pois ao "ver" uma imagem colorida, RGB, o que o computador tem é uma matriz de pixels cujos valores variam de 0 a 255 em três camadas, uma para cada cor.

Existem diferentes tipos de algoritmos de CNN, para este trabalho foi realizado o treinamento de uma RNA, descrito no Capítulo 3, utilizando a arquitetura *You Only Look Once* (YOLO), proposta por (REDMON; DIVVALA et al., 2016).

2.4.1 YOLO

É um algoritmo de detecção de objetos que utiliza CNN para detecção de objetos em tempo real (REDMON; DIVVALA et al., 2016) conhecido por sua capacidade de realizar detecção de objetos em uma única passagem, tornando-o mais rápido do que outros algoritmos como Faster R-CNN e SSD (Single Shot MultiBox Detector) (REDMON; DIVVALA et al., 2016; TAN et al., 2021). O modelo YOLOv3 é uma melhoria incremental em relação às versões anteriores, alcançando maior precisão e desempenho (REDMON; FARHADI, 2018), este algoritmo é de código aberto e está em constante atualização.

A principal vantagem do YOLO é seu desempenho em tempo real, permitindo a detecção rápida e eficiente de objetos. No entanto, é importante notar que a velocidade do YOLO tem o custo de uma precisão ligeiramente inferior em comparação com outros algoritmos (TAN et al., 2021).

Em resumo, YOLO é um algoritmo de detecção de objetos em tempo real que utiliza CNN para detecção rápida e eficiente de objetos. Tem sido aplicado em vários domínios e tem mostrado resultados promissores em termos de velocidade e desempenho, tornando-o atraente para a tarefa de classificação de defeitos de PCBs e PCBAs em uma linha de montagem.

Anotações do YOLO

As anotações do YOLO na Darknet são arquivos de texto em formato TXT com as informações de localização e a classe de um objeto em uma imagem. Para cada imagem utilizada para treinar um modelo de aprendizado de máquina usando YOLO há um arquivo de anotações contendo as informações sobre onde e qual classe dos objetos estão presentes na imagem. Cada linha no

arquivo de anotações corresponde a um objeto. O formato padrão de uma anotação é descrito a seguir:

YOLO DarkNet TXT

<object-class> <x> <y> <width> <height>

- **object-class:** É a classe, classificação, de um objeto. Para esta pesquisa um tipo de defeito.
- **x e y:** são as coordenadas do centro do objeto na imagem, normalizadas entre 0 e 1
- **width e height:** A largura e a altura do objeto na imagem, também normalizadas entre 0 e 1.

2.5 Placa de Circuito Impresso

A Placa de Circuito Impresso, PCI, também chamada de *Printed Circuit Board*, PCB, é uma placa fundamental para a fabricação de EEE pois através deste componente eletrônico que as conexões elétricas são facilitadas e a sustentação mecânica entre os demais componentes eletrônicos, *Surface Mounted Device* (SMD) e Sigla*Through-Hole Technology*THT, é realizada (KAYA; AKGÜL, 2022; RODRIGUES et al., 2019). Geralmente são feitas de material isolante, como por exemplo a fibra de vidro ou epóxi, e possui sobre suas superfícies trilhas condutoras em metal, normalmente o cobre que representa cerca de 27% da massa de uma PCB (YAMANE; ESPINOSA; TENÓRIO, 2013), responsáveis por conectar os componentes eletrônicos em um circuito (RODRIGUES et al., 2019). Embora existam estudos para substituição de sua composição visando a diminuição do impacto ambiental após o descarte desse tipo de material (LIU, J. et al., 2014).

A fixação de componentes SMD e THT utilizados na indústria eletrônica é realizada por meio de soldagem junto a PCB. Componentes SMDs costumam ter tamanho e massa reduzidos, adequados a projetos de miniaturização, e seus terminais são soldados diretamente na superfície da PCB o que torna a montagem mais rápida e eficiente por outro lado, componentes THTs requerem a inserção de seus terminais por meio de orifícios da PCB e soldagem na superfície oposta, podem ser maiores comparados aos SMDs, o que dificulta a miniaturização e design compacto. O uso deste tipo de componente requer um processo de soldagem maior e mais trabalhoso em comparação com SMDs e, consequentemente, mais

demorado. A escolha do tipo de componentes está intimamente ligada ao projeto de PCB (DEY; KUMAR, 2021).

Além dos componentes eletrônicos, as PCBs podem ser classificadas conforme suas características e métodos de fabricação. A escolha do tipo de PCB utilizada para produção de um EEE depende da complexidade do circuito, custo e restrição de espaço (HAQ; JILANI; PRABU, 2022). A seguir são listadas categorias de PCBs:

1. **PCBs de camada única:** Consiste em uma PCB com material condutor em uma única face, estão relacionadas a fabricação de EEE de consumo. Possuem limitações relacionadas ao espaço disponível (PERDIGONES; QUERO, 2022);

2. **PCBs de dupla camada:** São placas cujas duas faces possuem material condutor separadas por um substrato isolante, o que amplia as possibilidades de design e miniaturização (PERDIGONES; QUERO, 2022);

3. **PCBs multicamadas:** São placas semelhantes às expressas anteriormente, porém com camadas intermediárias, que permite um maior volume de circuitos em uma menor área (PERDIGONES; QUERO, 2022; BAGHERI et al., 2023);

Conforme sua aplicação, as PCBs também podem ser categorizadas mediante as necessidades de rigidez, flexibilidade e dissipação de calor exigida pela inserção desse componente em um EEE, expressos em outros estudos (LIU, J. et al., 2014; SELBMANN et al., 2023).

Após a inclusão e soldagem dos componentes sobre a PCB ela passa a ser reconhecida como *Printed Circuit Assembly*, PCBA, ou seja, uma placa funcional (FUNG; YUNG, 2020).

A fabricação de PCBAs para EEE em escala industrial é realizada por meio do uso de tecnologia *Surface-Mount Technology* (SMT) (FUNG; YUNG, 2020). Estima-se que 95% dos componentes de uma PCBA sejam SMDs (DEY; KUMAR, 2021), embora seja possível realizá-la de forma manual, conforme descrito no estudo (RODRIGUES et al., 2019).

A Seção a seguir aborda a tecnologia SMT, fatores históricos, vantagens e desvantagens e o processo de produção de PCBA.

2.6 Surface-Mount Technology

Na década de 80, o uso de SMT foi impulsionado como principal forma de produção de PCBAs. A SMT, conforme descrito na Seção anterior, envolve a criação de PCBAs cujos SMDs são soldados diretamente na superfície, garantindo um ganho de performance e agilidade da produção (BUENO et al., 2020). Atualmente a SMT é amplamente utilizada na fabricação de PCBAs para a produção dos mais diversos EEE. A crescente demanda por EEE menores, com o maior número de funções e cada vez mais eficientes, levou ao crescimento do uso de componentes menores também (MISTRY et al., 2022).

Para que haja sucesso na produção de EEEs menores, a indústria foi adequada ao desenvolvimento de novas PCBAs integradas a esses produtos. Aliado a esta produção os SMDs também tiveram seu custo reduzido (COMBET; CHANG, 2009). Figura 2.3 oferece a compreensão da redução de tamanho de resistores expresso no artigo (COMBET; CHANG, 2009) de 2004.

Utilizar componentes tão pequenos, Figura 2.3, para fabricação de PCBAs torna a produção e o reparo de PCBAs em escala, de forma manual, inviável (COMBET; CHANG, 2009).

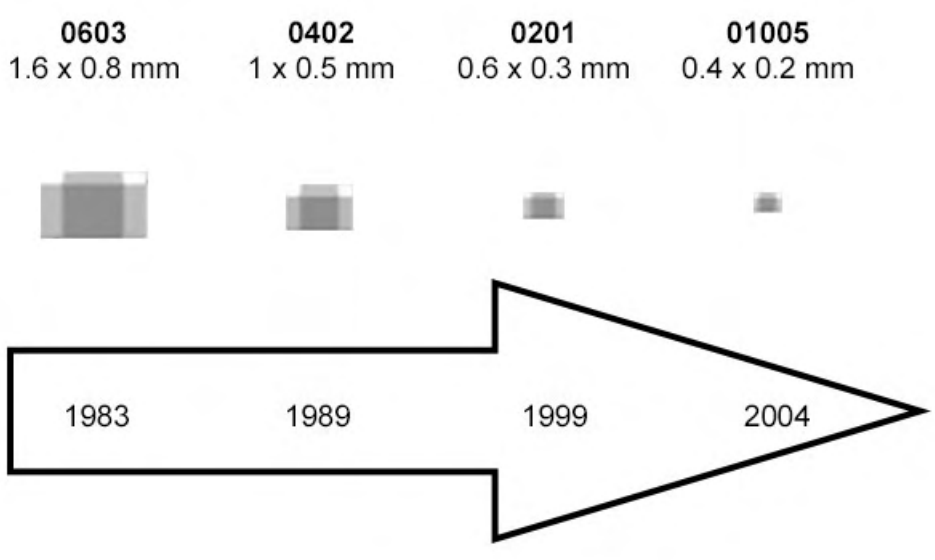


Figura 2.3: Evolução de resistor SMD ao longo dos anos - Adaptado de COMBET & CHANG, 2009.

Considerando a produção de PCBAs utilizando SMT, há redução de custo de produção a nível de recursos humanos e superfície de PCB. Tornando possível a criação de PCBAs

extremamente avançadas em conjuntos cada vez menores com repetibilidade inteligente graças ao seu nível de automação (DEY; KUMAR, 2021; HAQ; JILANI; PRABU, 2022).

2.6.1 Fluxos básicos de uma linha de produção utilizando a SMT

A linha de montagem com SMT transforma PCBs em PCBAs conforme descrito nos trabalhos (FUNG; YUNG, 2020; ASGHAR et al., 2019):

1. **Estação de Impressão:** A PCB é carregada no equipamento de impressão e posteriormente é submetida a impressão de pasta de solda. Nesta etapa a impressora utiliza um estêncil de aço inoxidável sobre a PCB. Acima do estêncil é depositada a pasta de solda que posteriormente é pressionada por uma espátula, que ao percorrer toda a extensão da placa deposita a pasta de solda sobre os *pads*, regiões da PCB destinadas a soldagem dos componentes (FUNG; YUNG, 2020; ASGHAR et al., 2019).

2. **Estação de montagem:** Através de máquinas *Pick-And-Place* os componentes SMDs são removidos da bobina de alimentação, *feeders*, posicionados sobre a superfície do PCB e pressionados, junto a posição calibrada previamente (RAMANATHAN et al., 2020).

3. **Estação de refusão:** Nesta estação a PCB é submetida ao forno de refluxo com objetivo de derreter a solda adicionada na impressão e posteriormente solidificar formando as juntas de solda e uma PCBA (FUNG; YUNG, 2020; ASGHAR et al., 2019)

Ao longo destes processos, defeitos de fabricação de PCBs e PCBAs podem ocorrer e impactar significativamente a qualidade e confiabilidade do produto. A Inspeção Óptica Automática, *Automatic Optical Inspection*, AOI é um método comumente utilizado para detecção de defeitos de forma não destrutiva. Os sistemas AOI usam câmeras e algoritmos de processamento de imagens para analisar as superfícies das PCBs e PCBAs com intuito de detectar defeitos, como defeitos de solda, componentes desalinhados, ausência de componentes e outras anomalias (BUENO et al., 2020). Ao comparar componentes reais da PCBA com componentes esperados no projeto, *golden sample*, os sistemas AOI detectam com rapidez e precisão defeitos que podem não ser facilmente visíveis ao olho humano. Com a identificação e sinalização de defeitos, os fabricantes podem tomar ações corretivas na linha de produção SMT.

2.7 Inspeção Óptica Automática

A Inspeção Óptica Automática é uma tecnologia utilizada para inspeção visual automatizada de produtos, componentes etc. Envolve o uso de sistemas ópticos de captura de imagens, sistema de iluminação e algoritmos de visão computacional para detectar e analisar defeitos e anormalidades cosméticas nos objetos inspecionados sem que seja necessário qualquer tipo de ação invasiva (NGUYEN; JENSSEN; ROVERSO, 2018).

Na indústria de fabricação de EEE, a AOI é utilizada para detectar erros visíveis em um estágio inicial da produção (SCHWEBIG, A. I. M.; TUTSCH, 2020).

Esta tecnologia desempenha um papel crucial na garantia da qualidade dos EEE antes da integração em seu ambiente funcional (SCHWEBIG, A. I. M.; TUTSCH, 2020).

Automatizar o processo de inspeção pode melhorar a eficiência e a confiabilidade do controle de qualidade durante a fabricação de Placas de Circuito Impresso, PCI em larga escala, além da maximização dos lucros (SCHWEBIG, A. I. M.; TUTSCH, 2020; SHAOBIN et al., 2022; CHOI et al., 2019).

No contexto do SMT, detalhado na próxima Seção, as estações AOI são normalmente integradas à linha de produção após os estágios de impressão da pasta de solda, estação de impressão, colocação de componentes, estação de montagem, e após a estação de refluxo.

Estas estações estão estrategicamente posicionadas para inspecionar a integridade das PCBs, as juntas de solda e componentes das PCBAs.

A primeira estação AOI geralmente está localizada após o processo de impressão da pasta de solda para inspecionar os depósitos de pasta de solda na PCB, garantir alinhamento, volume e ausência de defeitos, como pontes ou solda insuficiente. Comumente, esta estação de AOI é conhecida como *Solder Paste Inspection* (SPI) (MAR; YARLAGADDA; FOOKES, 2011; KIM et al., 2019).

A segunda estação AOI é normalmente posicionada após o processo de colocação do componente verificando seu alinhamento, presença e qualidade. Essa inspeção ajuda a identificar problemas como componentes desalinhados ou ausentes, marcações para exclusão, terminais levantados ou defeitos de solda (ZHAO; CHENG; JIN, 2009).

Já a terceira estação AOI é posicionada após a soldagem de refluxo, onde a PCBA já está formada. Neste momento as ligações elétricas devem estar em conformidade. Esta estação é super importante, pois estima-se que 43% do total de defeitos de uma PCBA ocorram neste momento (YAMANE; ESPINOSA; TENÓRIO, 2013).

Estações AOI adicionais podem ser incorporadas à linha de produção dependendo da complexidade e dos requisitos de qualidade da montagem do PCB. Essas estações podem ser colocadas em diferentes estágios do processo de montagem para realizar inspeções específicas, como verificação de polaridade, verificação da presença de componentes específicos ou inspeção de recursos especializados. É importante observar que o posicionamento exato e o número de estações AOI podem variar dependendo da configuração específica de fabricação e dos requisitos do processo SMT. Os fabricantes podem personalizar o posicionamento das estações AOI com base em fatores como volume de produção, complexidade do produto e objetivos de controle de qualidade.

Graças a sua performance, vantagens e confiança frente a processos tradicionais de inspeção a tecnologia AOI se tornou cada vez mais importante na tecnologia SMT (ZHAO; CHENG; JIN, 2009).

A Seção a seguir apresenta os tipos de defeitos ao qual a tecnologia AOI está apta a encontrar e classificar.

2.8 Defeitos de PCBs e PCBAs

A partir do uso da AOI podemos encontrar diversos defeitos em PCBs e PCBAs. A seguir a lista de 18 defeitos encontrados durante o levantamento bibliográfico:

1. **Missing Hole:** Buraco faltante é um problema comum em PCBs. O defeito ocorre quando não há um buraco na trilha que conecta componentes. Sua ausência impede a passagem de corrente e a fixação de componentes THT. Isso pode causar mau funcionamento ou falha no dispositivo. Comumente está associado a erros de fabricação ou uso inadequado da placa (WU; WANG, M.-J. J.; LIU, C.-M., 1996; CHAUDHARY; DAVE; UPLA, 2017; AKHATOVA, 2021);

2. **Mouse bite:** Uma mordida de rato em uma PCB normalmente se refere a uma pequena área localizada de dano ou a um pequeno buraco que é criado involuntariamente no substrato da PCB, também conhecida como deformação negativa. Esses danos podem ocorrer por diversos motivos, como estresse mecânico, manuseio inadequado ou erros de fabricação. O termo “mordida de rato” é usado para descrever a aparência do dano, que pode se assemelhar a uma pequena marca de mordida. Mordidas de rato podem ser problemáticas porque podem interromper as conexões elétricas na PCB, causando

circuitos abertos ou outros problemas elétricos. Esses defeitos podem afetar a funcionalidade e a confiabilidade da PCB e podem resultar na falha do EEE ou sistema no qual a PCB é usada. (WU; WANG, M.-J. J.; LIU, C.-M., 1996; CHAUDHARY; DAVE; UPLA, 2017; AKHATOVA, 2021);

3. **Circuito aberto:** Um defeito de circuito aberto em uma PCB ocorre quando há uma interrupção ou descontinuidade na trilha, conexão elétrica, que impede a corrente elétrica de fluir, como um furo, rachadura, arranhão ou quando há um componente defeituoso. O defeito pode ser difícil de detectar, pois pode não apresentar sinal visível para indicar a localização do problema, sendo necessário aplicar um teste de continuidade (WU; WANG, M.-J. J.; LIU, C.-M., 1996; CHAUDHARY; DAVE; UPLA, 2017; AKHATOVA, 2021);

4. **Curto-circuito:** Um curto-circuito é um defeito de circuito elétrico que ocorre quando dois ou mais condutores são conectados diretamente, permitindo que a corrente elétrica flua entre eles. Isso geralmente acontece quando há um problema com o circuito, como um fio solto ou uma trilha de conexão indevida na PCB. Quando isso acontece, a corrente elétrica pode fluir em excesso, causando superaquecimento, danos à placa e aos componentes conectados. Um curto-circuito em uma PCB pode ser causado por diversos fatores, incluindo falhas na fabricação da PCB, erros de montagem dos componentes e intempéries. Existem várias técnicas que podem ser usadas para detectar curtos-circuitos em PCBs, incluindo testes de continuidade, testes de resistência e testes de tensão. Além disso, as ferramentas de design de PCBs modernas podem ajudar a identificar possíveis problemas durante o projeto da placa (WU; WANG, M.-J. J.; LIU, C.-M., 1996; CHAUDHARY; DAVE; UPLA, 2017; AKHATOVA, 2021);

5. **Spur e Spurious Cooper:** Solda despejada indevidamente sobre regiões da PCB em inconformidade com o design original do projeto de PCB. Podem ocorrer por diversas causas como corrosão inadequada, soldagem inadequada, contaminação etc. Este tipo de defeito também é conhecido como deformação positiva (WU; WANG, M.-J. J.; LIU, C.-M., 1996; CHAUDHARY; DAVE; UPLA, 2017; AKHATOVA, 2021);

Defeitos não mapeados pelo *dataset* utilizado no experimento

6. **Wrong size hole:** Furo com diâmetro inadequado para fixação de componentes THT (CHAUDHARY; DAVE; UPLA, 2017). Podem causar a não fixação caso o diâmetro seja menor que o esperado ou a soltura do componente durante a manipulação caso o diâmetro seja maior.

7. **Falta de solda:** Defeito na PCB sem solda ocorre quando a solda não adere adequadamente à superfície metálica do componente ou a PCB. Isso pode fazer com que as conexões elétricas não sejam confiáveis ou até mesmo falhem completamente.

8. **Pin-Hole:** Buraco na trilha é um problema comum em PCBs. O defeito ocorre quando há um buraco na trilha que conecta dois componentes, o que impede a passagem de corrente idealizada na fase de projeto. Isso pode causar mau funcionamento ou falha no dispositivo. O defeito pode ser causado por vários fatores, incluindo erros de fabricação, uso inadequado da placa, exposição a temperaturas extremas ou a umidade excessiva.

9. **Solda insuficiente:** Defeito causado por solda insuficiente ocorre quando a solda usada para conectar os componentes na PCB não tem calor ou tempo suficiente para derreter adequadamente e formar uma ligação forte ou há pouco material de solda. Tem consequências similares à falta de solda.

10. **Excesso de solda:** Este defeito ocorre quando muita solda é aplicada à PCB. Isso pode causar curtos elétricos, conexões ruins e outros problemas.

11. **Lifted lead:** Defeito em uma PCBA causado pela separação indevida do terminal do componente com o terminal da PCB. Isso pode ser causado por soldagem inadequada, choque mecânico ou ciclagem térmica. Este defeito pode causar curtos-circuitos elétricos e circuitos abertos, resultando em mau funcionamento do PCBA.

12. **Componente ausente:** Um componente ausente em uma PCBA pode causar uma variedade de problemas. Dependendo do tipo de componente, pode causar mau funcionamento da PCBA ou até falha total. Em alguns casos, um componente ausente pode levar a um circuito aberto, o que pode causar superaquecimento ou instabilidade da placa. Além disso, se o componente ausente fizer parte de um circuito ativo, pode impedir que o circuito funcione corretamente.

13. **Componente errado:** Ocorre quando um componente de uma PCBA não corresponde às especificações de design do projeto. Isso pode ocorrer devido a vários motivos, como

compra incorreta, instalação inadequada ou uso incorreto de um componente. Em alguns casos, o componente errado pode ser instalado no local errado da PCB. Isso pode levar a curtos-circuitos ou outros problemas de funcionamento. Para evitar esse problema, é importante garantir que todos os componentes sejam solicitados e instalados corretamente de acordo com as especificações do projeto. Além disso, é importante verificar se todos os componentes são compatíveis entre si antes da instalação.

14. **Componente danificado:** Um componente danificado é um componente que foi fisicamente danificado de alguma forma e, portanto, não executa suas funções no circuito adequadamente.

15. **Componente invertido:** Defeito de PCB envolvendo um componente invertido ocorre quando o componente é soldado na orientação errada. Isso pode fazer com que o componente não funcione corretamente ou até mesmo danificar a PCB. Pode causar curtos ou outros problemas elétricos.

16. **Componente com polaridade invertida:** Um defeito de polaridade invertida ocorre quando os terminais positivo e negativo de um componente são invertidos. Isso pode causar danos a PCB como superaquecimento e, em casos mais severos, incêndio e explosão.

17. **Shifted:** Um componente deslocado é um defeito que ocorre quando um componente da PCB não está alinhado corretamente com sua posição designada. Isso pode ser causado por soldagem inadequada, posicionamento incorreto do componente ou defeito de fabricação. Componentes deslocados podem causar curtos-circuitos, circuitos abertos e outros problemas que podem levar ao mau funcionamento da PCB.

18. **Tombstone/Bilboard:** Um tipo de defeito de PCBA que ocorre quando um componente parece estar em pé, devido ao torque realizado pela solda. Esse tipo de defeito pode ocorrer quando os componentes não estão alinhados corretamente durante o processo de soldagem ou quando os componentes são colocados muito próximos uns dos outros.

Capítulo 3

Desenvolvimento

Este capítulo descreve em detalhes as atividades desenvolvidas nesta pesquisa em prol da criação de uma proposta de software gerador de *datasets* de PCIs. Há também o treinamento de um modelo de aprendizado de máquina utilizado para detecção de defeitos em artefatos gerados pela proposta de *software*.

3.1 Materiais e Métodos

Durante a realização desta dissertação foram utilizadas pesquisas bibliográficas, revistas acadêmicas, livros e artigos científicos disponíveis *online* para a fundamentação teórica e contextualização do domínio.

A proposta de *software* é detalhada por meio de diagramas de caso de uso, requisitos funcionais, protótipos de tela e arquitetura.

Os protótipos de tela foram desenvolvidos através da do *Integrated Development Environment* (IDE), *Microsoft Visual Studio 2022 Community*, para aplicações do tipo *Windows Forms*, aplicações *desktop*.

Para provar conceitualmente o funcionamento do *dataset* gerado pela proposta. Foi realizado o treinamento de um modelo utilizando YOLO com o *dataset* público *PCB Defects* - (AKHATOVA, 2021). Este modelo não contém todos os defeitos levantados durante a fase de levantamento bibliográfico, em consonância com o problema ao qual esta proposta busca resolver, porém através dele é possível validar que o *software* proposto gera informações em conformidade com o uso para as atividades de treinamento, validação e testes. Ao considerar

que todos os artefatos de saída gerados pela proposta seguem o mesmo padrão do *dataset PCB Defects* utilizado como entrada.

O YOLO foi utilizado por estar aderente ao esperado em uma linha de produção, velocidade de detecção e classificação, em razão do formato de produção dos EEE.

Para finalizar foram realizadas detecções sobre o *dataset* gerado pela proposta.

3.2 Proposta de *Software* Gerador de Placa de Circuito Impresso

Este trabalho propõe a criação de um *software* gerador de *dataset* de PCIs sintéticas, análogo aos utilizados para criação de modelos de visão computacional utilizados em estações de AOI. Portanto, é necessário um conjunto de funcionalidades, cujos requisitos estão na Seção 3.2.2, que suportem a criação de imagens e anotações, necessárias para o treinamento, teste e validação de um modelo e posterior detecção de defeitos em PCIs. Neste contexto PCBs, compreendem PCBs e PCBAs.

Para criar um *dataset* esta proposta tem uma série de funcionalidades dispostas no diagrama de caso de uso, Seção 3.2.1, que uma vez realizadas permitem a criação de um dataset. O fluxo básico de funcionamento, Figura 3.1, do *software* é compreendido nas seguintes ações:

1. **Cadastrar Componentes:** Funcionalidade responsável pela inclusão do nome do componente e suas respectivas imagens, com e sem defeitos;

2. **Cadastrar PCB:** Funcionalidade responsável pela inclusão do nome da PCB e inclusão da imagem da PCB;

3. **Associar PCB e Componentes:** Sub-rotina do Cadastro de PCB onde os componentes são incluídos e posicionados sobre a imagem da PCB;

4. **Gerar Dataset:** Funcionalidade responsável pela combinação de imagens dos componentes sobre a imagem da PCB. Além da criação dos arquivos de imagem e anotações no formato YOLO DarkNet TXT;

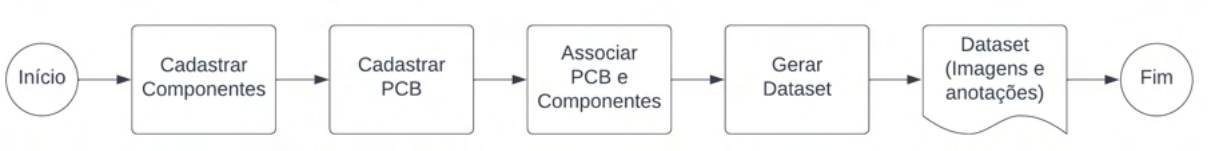


Figura 3.1: Fluxo Básico de funcionamento do *software*

Em resumo, o *software* utiliza como informações de entrada a imagem da placa, componentes sem defeito, componentes com defeito e as posições de alocação de cada componente. A partir dessas informações, o *software* realiza combinações de componentes sobre a placa. Usando a sobreposição das imagens de componentes na placa, o *software* retorna como saída imagens, representações de PCIs, cujos nomes contém identificação única e auto incrementável que em conjunto com os arquivo de anotações compõem o *dataset*.

3.2.1 Diagrama de Caso de Uso

O Diagrama de Caso de Uso, Figura 3.2, tem como finalidade ilustrar os requisitos funcionais de um sistema descrevendo as iterações entre os atores e a aplicação, expondo os diferentes cenários possíveis (FOWLER, 2006). Para esta proposta existem dois atores:

1. O usuário, responsável por utilizar a interface visual e manualmente preencher os cadastros de componentes e PCBs

2. *Pick-and-Place Machine*, partindo da premissa que os componentes são posicionados sobre a placa antes da inspeção óptica automática, uma interface de comunicação pode ser criada e implementada, para preencher os dados de componentes e posição de uma PCB, conforme a calibração desta máquina.

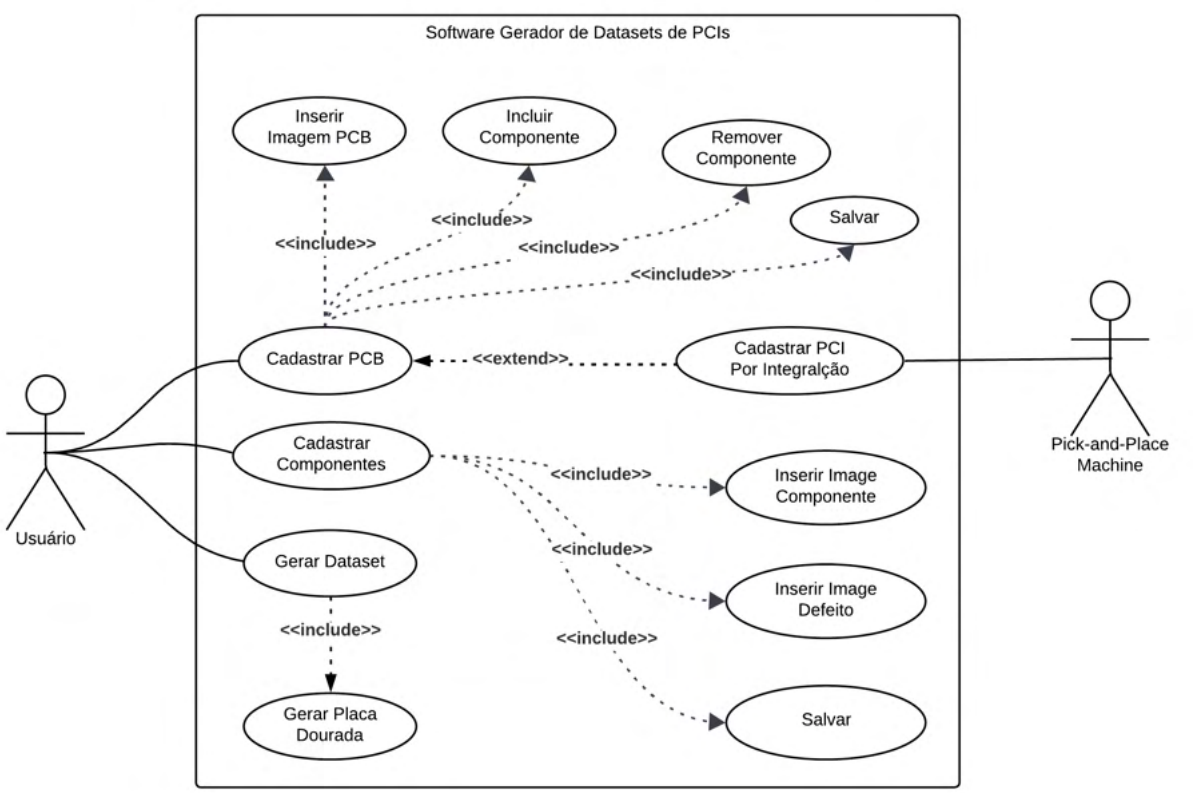


Figura 3.2: Diagrama de Caso de Uso - *Software* Gerador de Dataset de PCIs.

Este diagrama contém uma visão macro de possibilidades de uso do *software*, os detalhes de cada uma das funcionalidades estão na Seção seguinte.

3.2.2 Requisitos Funcionais

A proposta de *software* contém três artefatos, objetos, principais que representam as entidades da aplicação: os Defeitos, os Componentes e as PCBs. A interação entre estes objetos é feita por meio das funcionalidades descritas nas Subseções a seguir.

Defeitos

1. Como pré-requisito os defeitos levantados no Capítulo 2, Seção 2.8, devem estar cadastrados em banco de dados;

2. O *software* deve listar os defeitos para realização do vínculo de Componentes com Defeitos;

Componentes

1. O *software* deve cadastrar componentes em banco de dados para que posteriormente os mesmos possam ser utilizados sobre as PCBs;
    - O cadastro é feito a partir do nome do componente, esta informação identifica o componente em toda a aplicação;
    - Ao realizar o cadastro, a inclusão de imagens de componentes com e sem defeitos ocorre;
    - As imagens de defeitos são cadastradas a partir do vínculo a um defeito previamente cadastrado;
2. O *software* deve listar os componentes cadastrados para que os mesmos possam ser editados ou utilizados ao serem referenciados por PCBs;
3. O *software* deve permitir a edição dos componentes com relação as suas imagens;

Para o Caso de Uso "Cadastrar Componentes", foi gerado o diagrama de atividades, Figura 3.3, e também os protótipos de telas, Figura 3.4 e Figura 3.5 com indicações em vermelho, conforme a lista enumerada:

1. O usuário informa o nome do componente;
2. Insere a imagem sem defeito dos componentes;
3. Seleciona o defeito dentre as versões com defeito e insere a imagem com defeito;
4. O passo 3 ocorre quantas vezes for necessário, para que diversos defeitos possam ser vinculados a um mesmo componente;
5. Salva a edição do componente

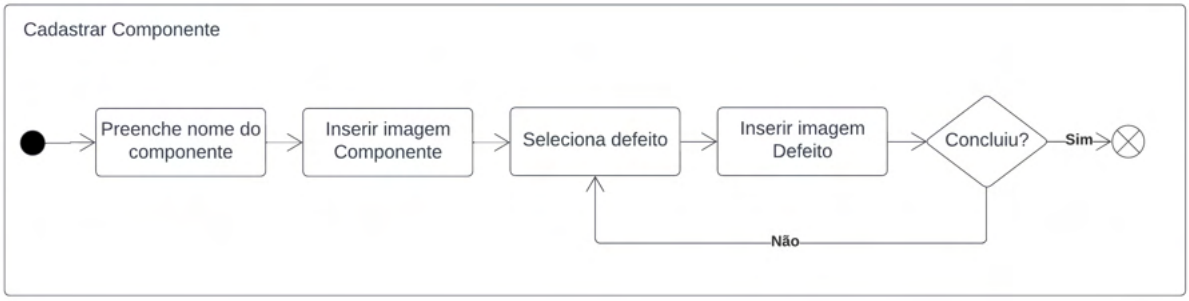


Figura 3.3: Diagrama de atividades - Cadastro de Componente.

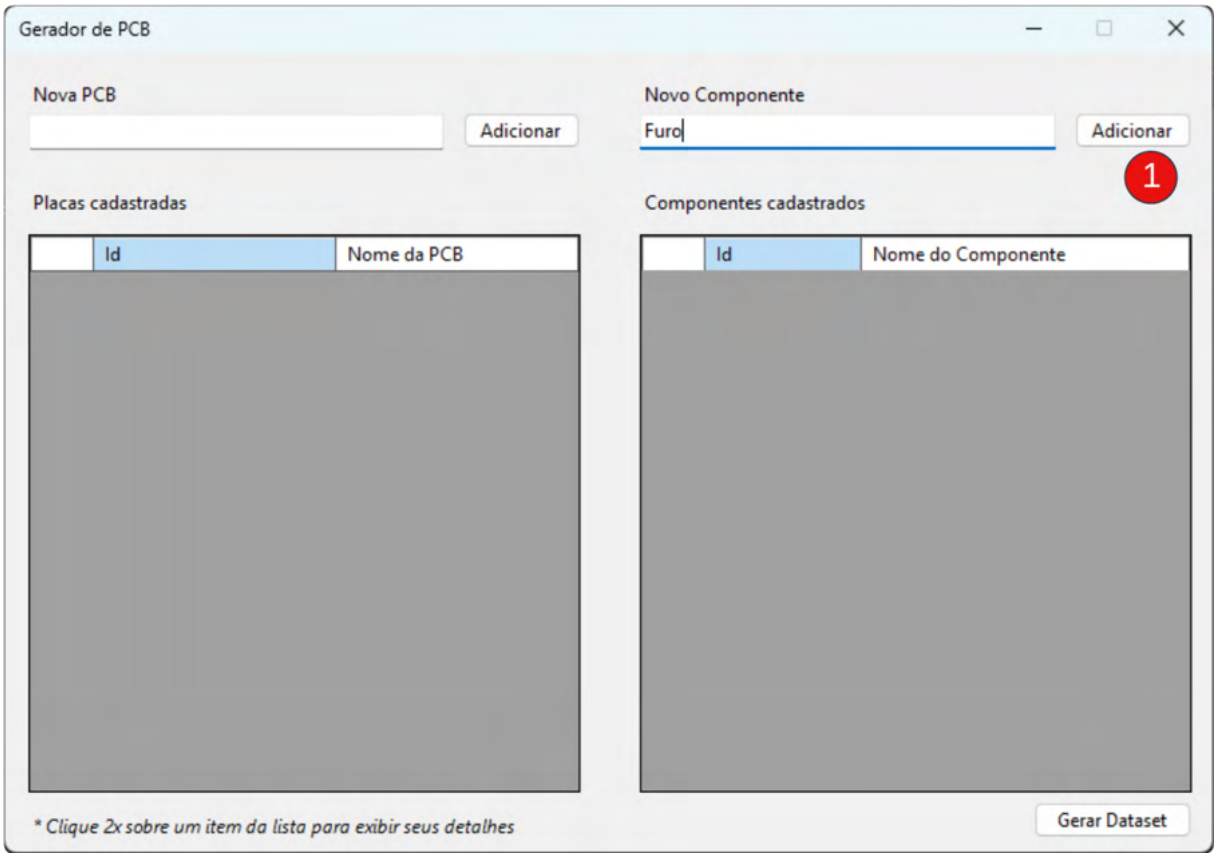


Figura 3.4: Protótipo de Tela - Tela inicial da aplicação - Inclusão de Componente.

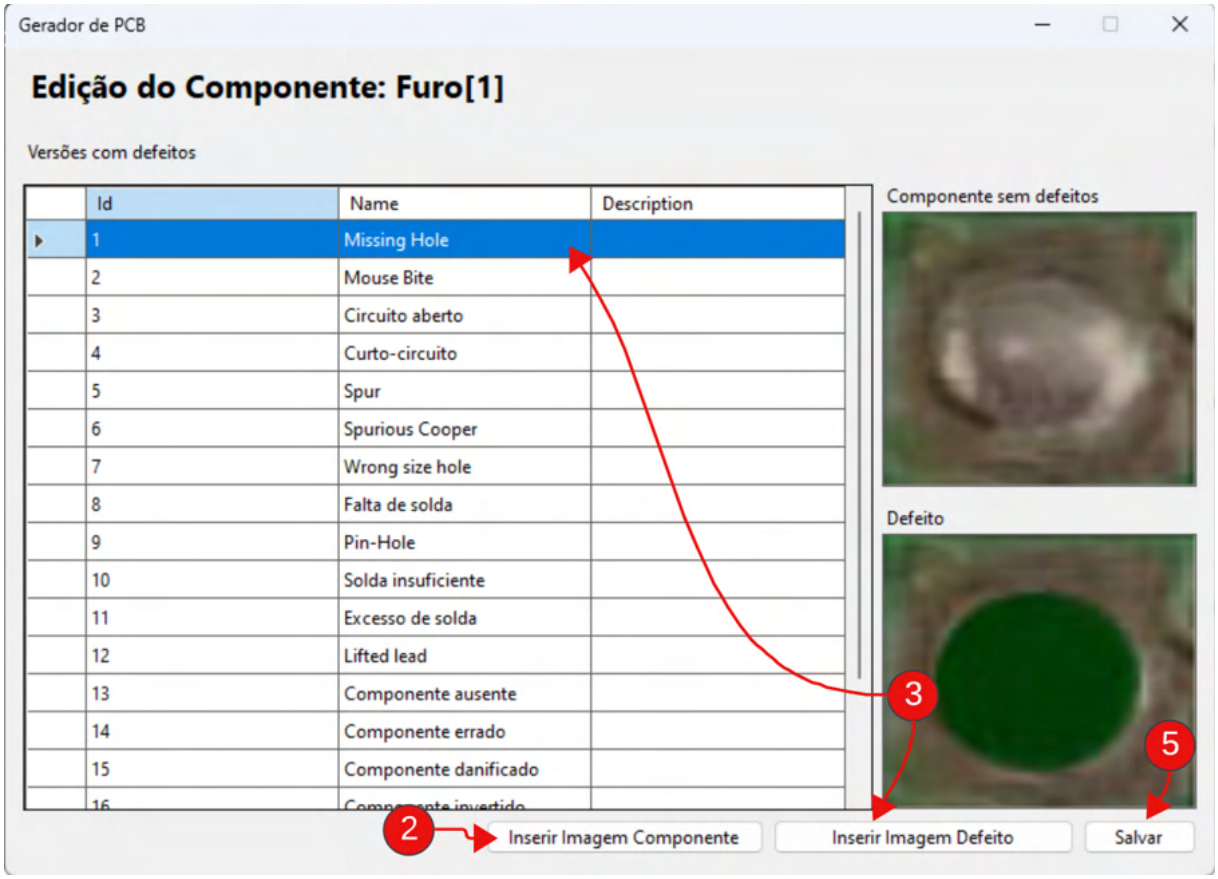


Figura 3.5: Protótipo de Tela - Edição de Componente.

PCB

1. O *software* deve cadastrar PCBs em banco de dados;
   (a) O cadastro é feito a partir do nome da PCB, esta informação serve como referência a PCB em toda a aplicação;
2. Permitir a inclusão ou sobrescrita de uma imagem de PCB;
3. Os componentes cadastrados devem ser listados em forma de componentes disponíveis para inserção sobre a PCB;
4. O *software* deve incluir componentes cadastrados sobre a PCB conforme posicionamento informado pelo usuário;
5. Os componentes incluídos devem ser listados para que possam ser reposicionados sobre a PCB ou selecionados

6. O *software* deve remover componentes incluídos e selecionados que estejam incorretos;
7. O *software* deve listar as PCBs cadastradas para futuras edições ou gerações de *datasets*;
8. O *software* deve permitir a edição das PCBs com relação a seus componentes;

Para o Caso de Uso "Cadastrar PCB", também foi gerado o diagrama de atividades, Figura 3.6, e protótipos de telas, Figura 3.7 a Figura 3.12 com indicações em vermelho, conforme a lista enumerada:

1. O usuário informa o nome da PCB;
2. Insere a imagem da PCB
3. Salva a inserção da imagem, esta ação é necessária para que o vínculo dos componentes aconteça na sequência;
4. O usuário insere componentes da lista de componentes disponíveis. Ao inserir o componente aparece sobreposto a imagem selecionada na ação anterior;
5. Os componentes inseridos passam a existir na lista de componentes incluídos;
6. O usuário posiciona, arrastando e soltando, o componente no local adequado;
7. O usuário repete os passos 3, 4 e 5 quantas vezes forem necessárias até que seus critérios sejam satisfeitos;
8. Para finalizar a PCB o usuário realiza a ação salva;

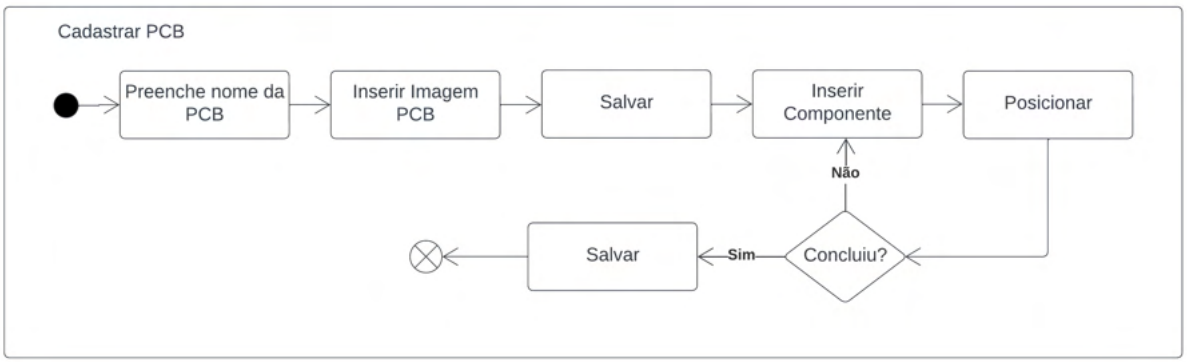


Figura 3.6: Diagrama de atividades - Cadastro de PCB.

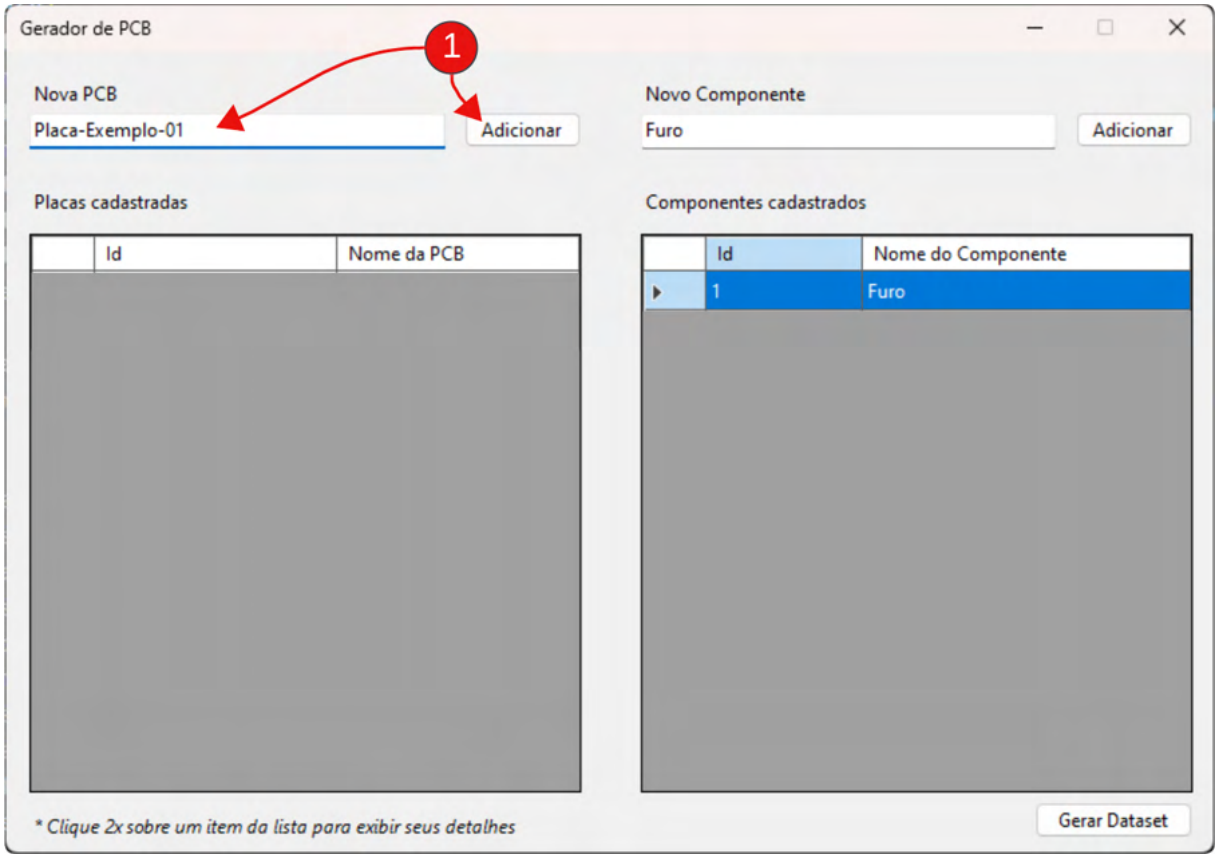


Figura 3.7: Protótipo de Tela - Tela inicial da aplicação - Inclusão de PCB

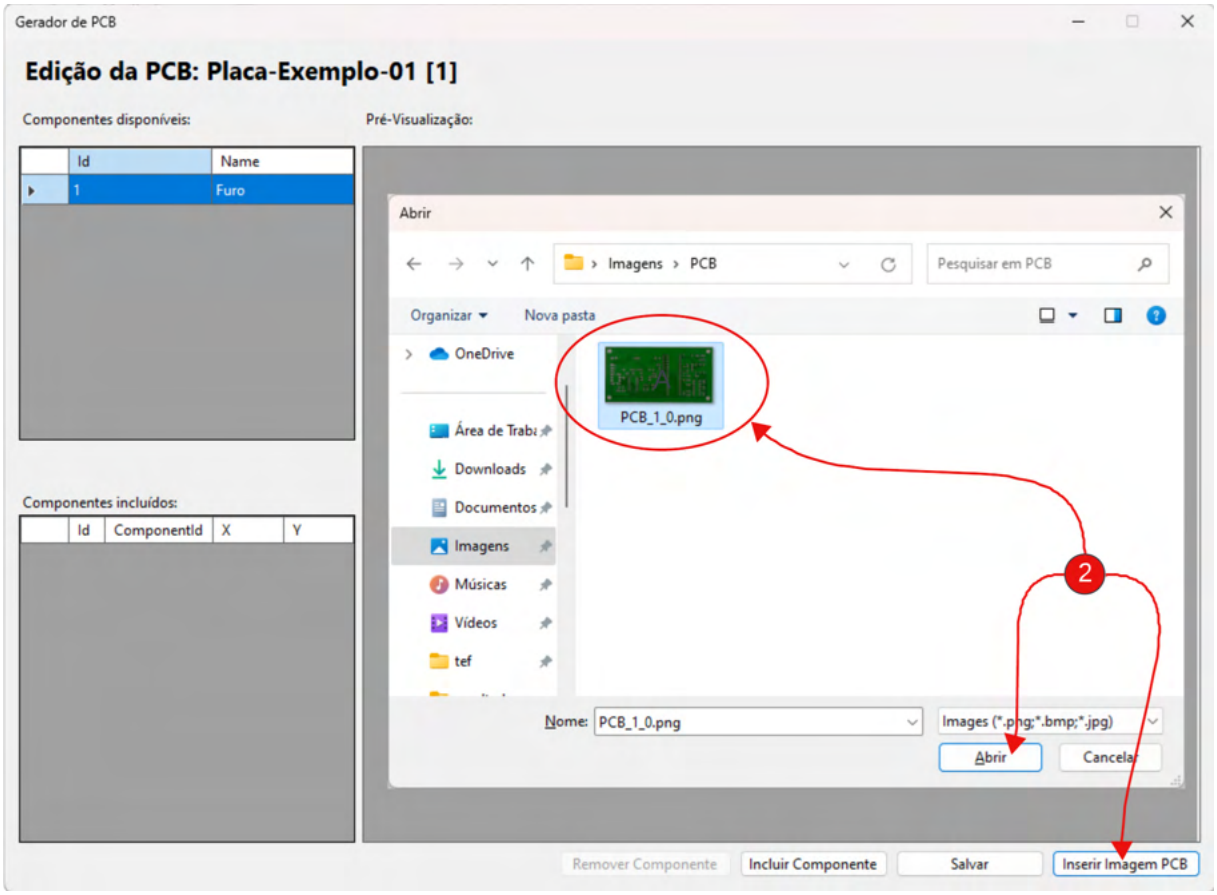


Figura 3.8: Protótipo de Tela - Edição de PCB - Inclusão de imagem de PCB

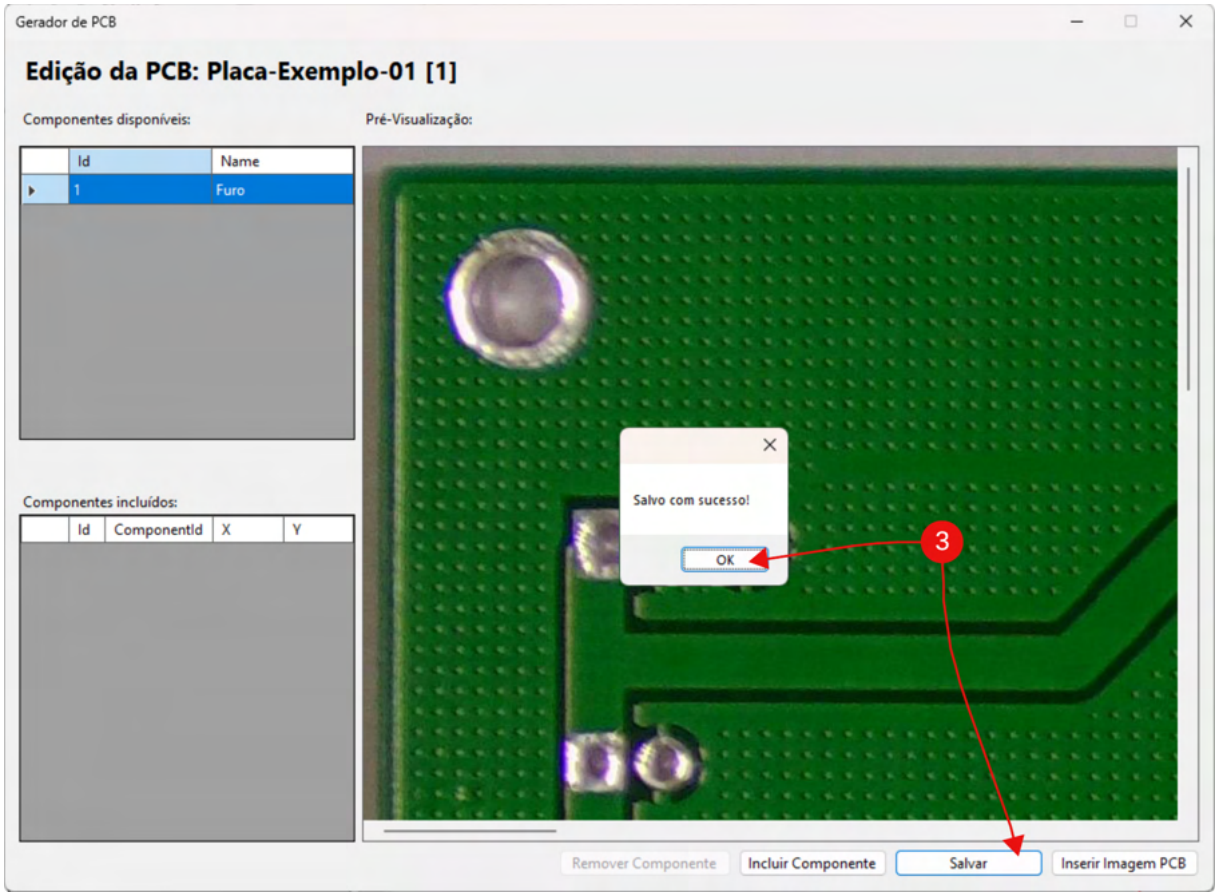


Figura 3.9: Protótipo de Tela - Edição de PCB - Salvando imagem de PCB

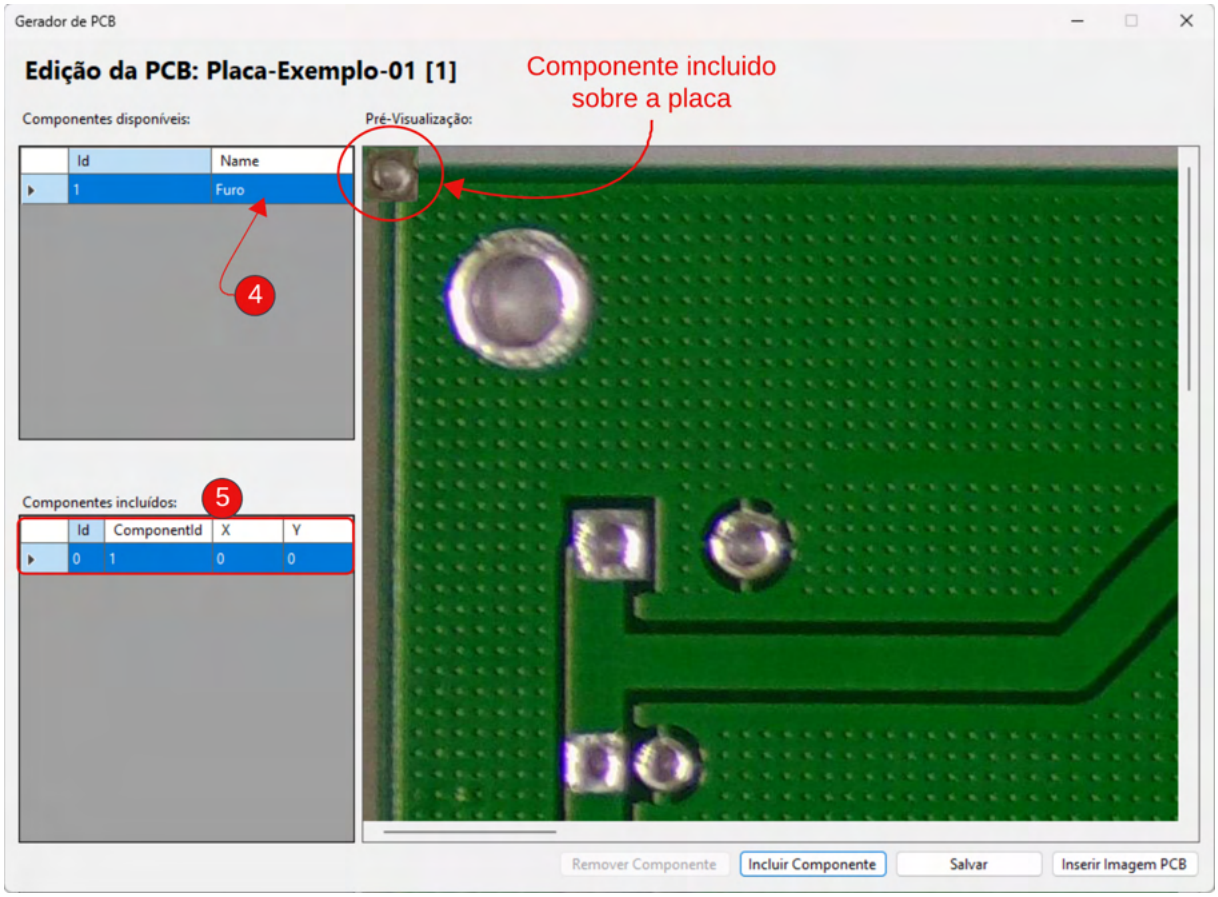


Figura 3.10: Protótipo de Tela - Edição de PCB - Incluindo Componente

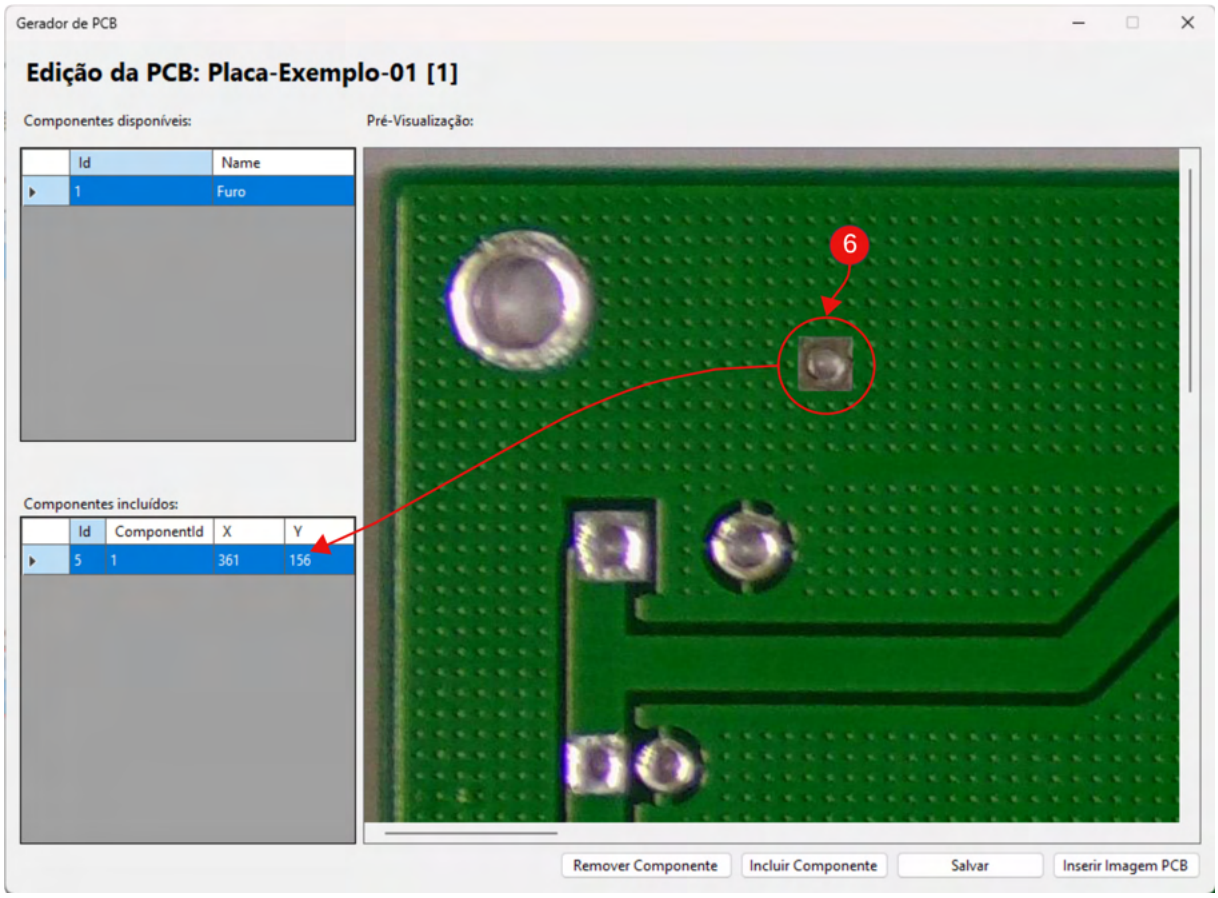


Figura 3.11: Protótipo de Tela - Edição de PCB - Posicionando Componente

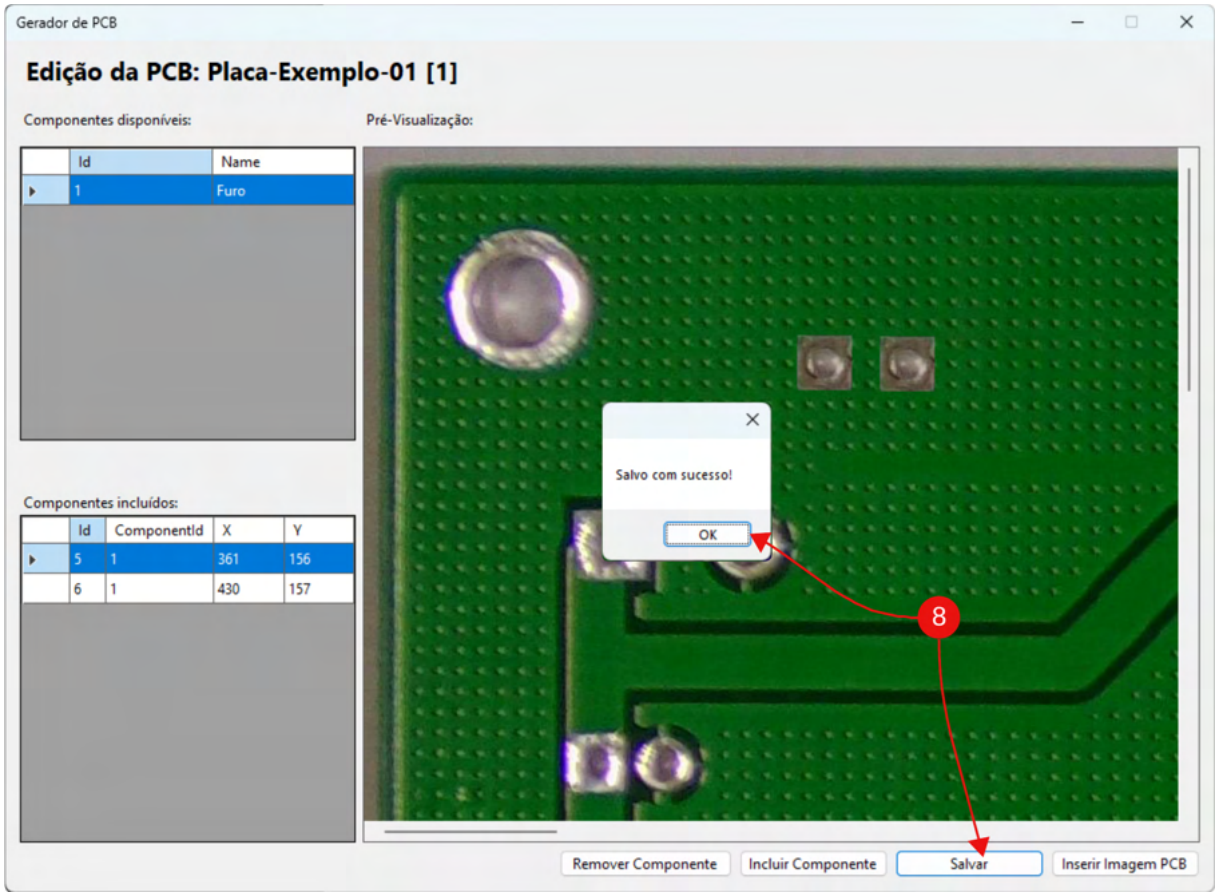


Figura 3.12: Protótipo de Tela - Salvando a PCB

Concluindo esta etapa é possível gerar o *dataset* conforme o que foi cadastrado previamente, tomando como exemplo a Placa-Exemplo-01 presente em todo o processo de criação, conforme protótipo, Figura 3.12.

Gerar *Dataset*

1. Pré-requisitos: PCBs com imagem de fundo definida
2. O *software* deve gerar arquivos de imagem e arquivos de anotação conforme o prévio cadastro da PCB e combinações possíveis;
3. A imagem cujo arquivo de anotação estiver vazio é a placa dourada, ou seja, placa sem defeitos;

Esta funcionalidade ocorre conforme o diagrama de atividades , Figura 3.13. Ao Considerando que durante a construção do protótipo um componente "Furo", Figura 3.5, com uma imagem de componente e um defeito estão cadastrados, e a placa "Placa-Exemplo-01", Figura 3.12, foi gerada com dois componentes "Furo". A combinação simples irá resultar em 4 arquivos de imagem e 4 arquivos de anotação salvos no diretório informado pelo usuário, Figura 3.14. Esta saída pode aumentar quando novas imagens de defeito e componentes são incluídas, resultando em novas combinação simples.

Figura 3.13: Diagrama de atividades - Gerar Dataset.

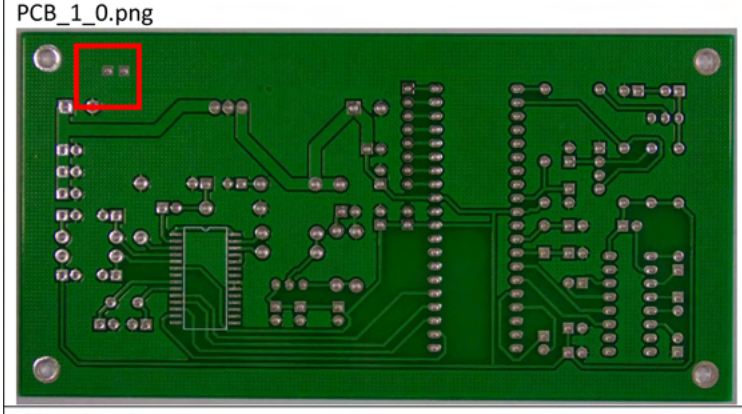
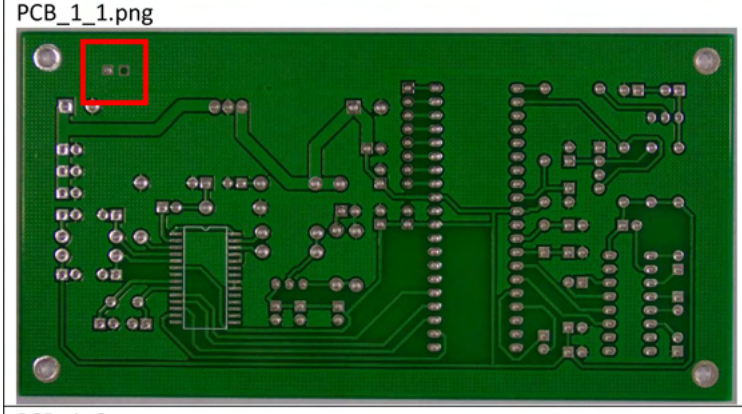
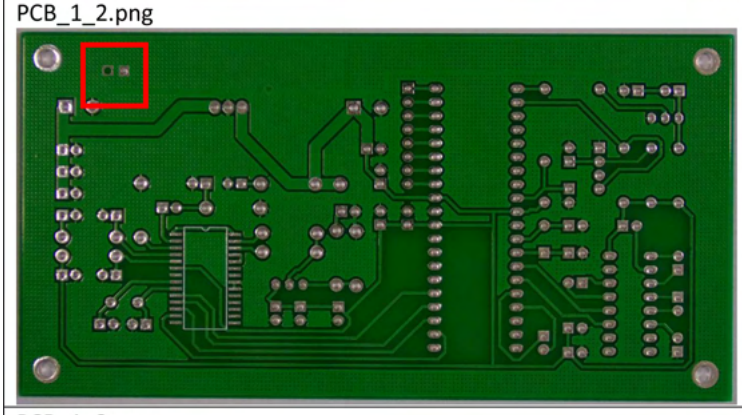
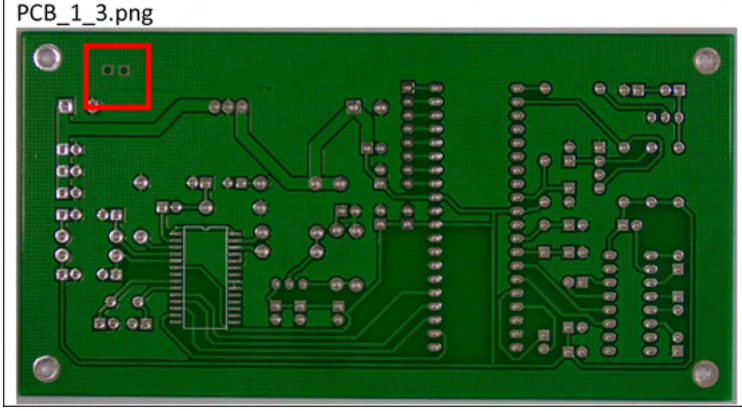
Imagens	Anotações
PCB_1_0.png 	PCB_1_0.txt: Vazio
PCB_1_1.png 	PCB_1_1.txt: 0 0.14930784442979564 0.11349306431273644 0.015161502966381015 0.029003783102143757
PCB_1_2.png 	PCB_1_2.txt: 0 0.12656558998022413 0.1128625472887768 0.015161502966381015 0.029003783102143757
PCB_1_3.png 	PCB_1_3.txt: 0 0.12656558998022413 0.1128625472887768 0.015161502966381015 0.029003783102143757 0 0.14930784442979564 0.11349306431273644 0.015161502966381015 0.029003783102143757

Figura 3.14: Resultado da Geração de Dataset.

3.3 Arquitetura do *Software*

A arquitetura de *software* refere-se à organização de um sistema, compreendendo seus componentes e seus relacionamentos. É responsável por fornece um modelo para a construção e manutenção de *softwares*, promovendo a comunicação entre as partes interessadas e orientando o design de alto nível do sistema.

Para esta proposta o modelo de arquitetura é o de camadas, *Model View Control* (MVC). Esta arquitetura é um padrão de design simples e amplamente utilizado no desenvolvimento de *software*. Este padrão propõe a divisão da lógica de apresentação e negócios em três componentes principais (DING et al., 2014):

- O *Model* representa os dados e a lógica de negócios da aplicação.
- A *View* é responsável pela apresentação dos dados ao usuário
- O *Controller* atua como intermediário entre o Modelo e a Visão, gerenciando as entradas do usuário e atualizando o Modelo conforme necessário.

A forma como as classes foram criadas e a comunicação entre elas estão na Figura 3.15

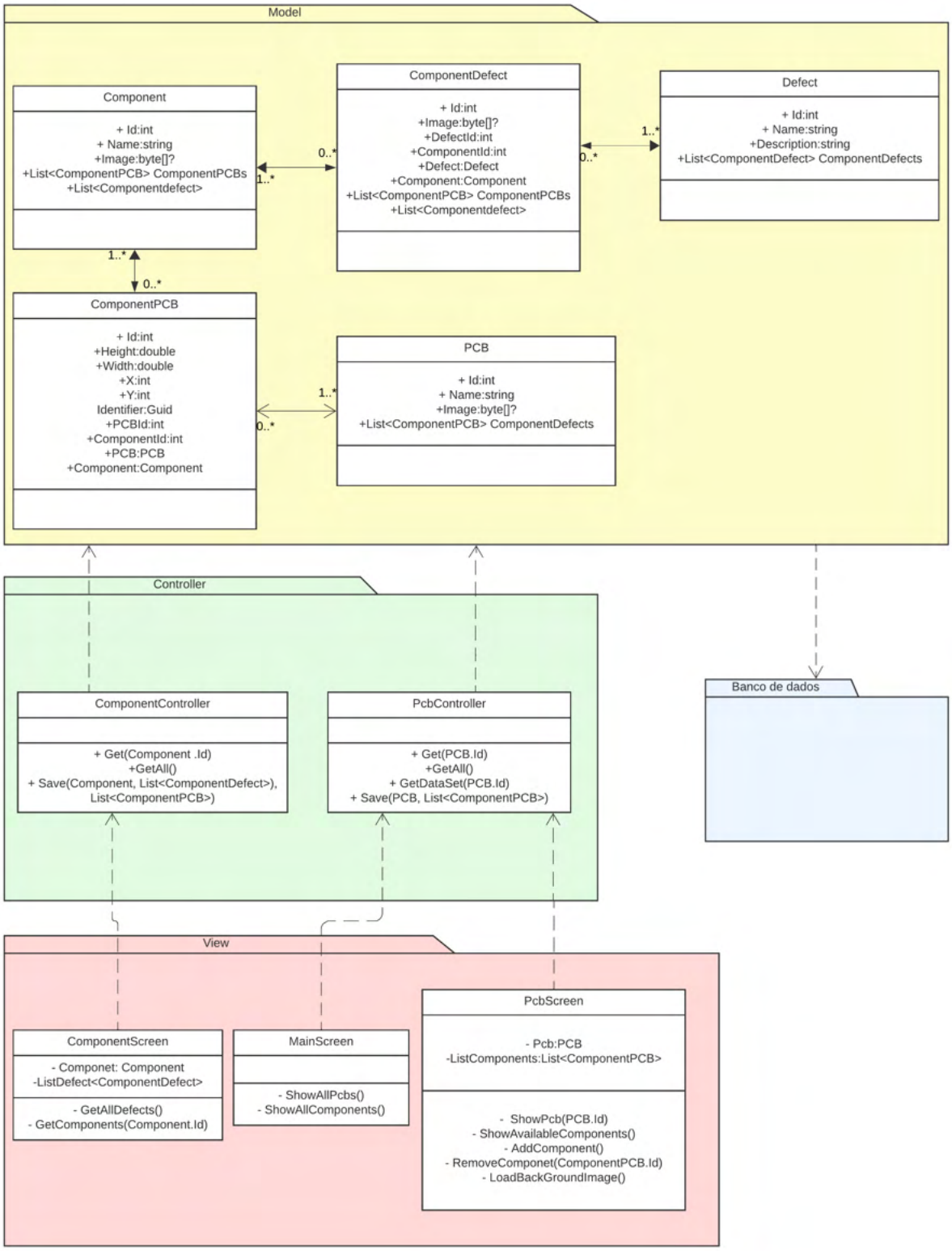


Figura 3.15: Diagrama de classes da proposta.

A opção por esta arquitetura tem como objetivo oferecer seus benefícios devido a sua simplicidade, separação clara de responsabilidades, que facilita a manutenção e a evolução do

software. Além disso, promove a reutilização de código e a escalabilidade do sistema. A separação de preocupações fornecidas pelo MVC também facilita a colaboração entre equipes de desenvolvimento, uma vez que diferentes aspectos da aplicação podem ser trabalhos de forma independente. O padrão MVC é amplamente utilizado em várias áreas, como o desenvolvimento de aplicações web, aplicativos móveis e sistemas de software em geral.

3.3.1 Model

Esta camada trata das classes que interagem com o banco de dados. Para esta proposta temos as seguintes classes:

1. *Component:* Representação dos componentes da PCB;
2. *Defect:* Representação dos defeitos;
3. *ComponentDefect:* Relacionamento da classe Component com a Classe Defect
4. *PCB:* Representação da PCB
5. *ComponentPCB:* Relacionamento da Classe PCB e seus Componentes;

3.3.2 View

Camada responsável por exibir os formulários da aplicação e manipular os dados antes da ação se salvar. Ao todo são 3 formulários.

1. *ComponentScreen*: Interface para manipulação de componentes;
2. *PcbScreen*: Interface para manipulação de PCBs;
3. *MainScreen*: Tela inicial da aplicação;

3.3.3 Controller

Camada entre as camadas *View* e *Model*, responsavel por implementar processamentos e validações, quando necessário.

3.4 Treinamento do Modelo

Para provar o conceito de que o *software* proposto produz *dataset* válido, cujas informações estejam adequadas ao uso em modelos de visão computacional, foi realizado um treinamento de aprendizado máquina com o *dataset* publico *PCB Defects*, (AKHATOVA, 2021), utilizando a rede neural convolucional YOLO, no formato Yolov5s. Após o treinamento do modelo, imagens do *dataset* produzidos pela proposta de *software* são utilizadas na atividade de detecção. Para esta abordagem os defeitos do *dataset* produzido pela proposta de *software* estão limitados aos classificadores do *dataset PCB Defects*. Esta limitação acontece justamente por conta do problema ao qual esta pesquisa pretende resolver uma vez que *datasets* novos, ainda que sintéticos, possam ser produzidos.

A seguir estão os passos adotados para criação do modelo.

3.4.1 Conjunto de dados (*Dataset*)

O conjunto de dados *PCB Defects*, (AKHATOVA, 2021), é composto por 690 arquivos de imagens, extensão JPG, somados a 690 arquivos de anotações, extensão XML, de localização e classificação dos 6 tipos de defeitos categorizados. Contém um total de 2940 anotações, balanceadas conforme Figura 3.16, com média aproximada de 4,3 categorias por arquivo de imagem. Está disponibilizado conforme Tabela 3.1.

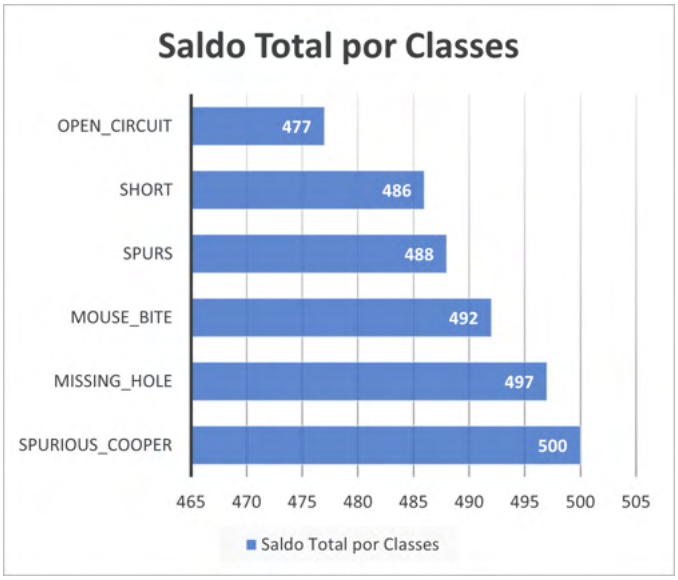


Figura 3.16: Saldo total por classes do conjunto de dados.

Classificadores PCB Defects		
Identificação Numérica	Classificação do defeito	Descrição
1	missing hole	Furo faltante para componentes THT ou fixação
2	mouse bite	Trecho de trilha do circuito estreitado indevidamente
3	open circuit	Trilhas do circuito interrompidas indevidamente.
4	short	Trilhas do circuito em curto, fechadas indevidamente
5	spur	Esporão, trecho de trilha expandida indevidamente
6	spurious copper	Traços ou áreas de cobre não intencionais

Tabela 3.1: Classificadores PCB Defects.

3.4.2 Pré-processamento

Os arquivos de anotações estão no formato VOC PASCAL XML e foram convertidos para o formato YOLO DarkNet TXT. Para realização da conversão foi utilizado o *Software as a Service*, Saas, (ROBOFLOW, 2023), o Roboflow é uma plataforma que dispõe de diversos recursos que auxiliam no desenvolvimento de modelos de visão computacional.

Além da conversão, os arquivos de imagem e anotações DarkNet foram enviados para a plataforma. Na sequência do pré-processamento do *dataset* os arquivos, de imagem e anotação, foram separados na proporção: 70% para treinamento, 20% para validação e 10% para teste, ou seja 483, 138 e 69 pares de arquivos. Dos quais têm, internamente, a frequência de de classes conforme Figura 3.17, Figura 3.18 e Figura 3.19 respectivamente.

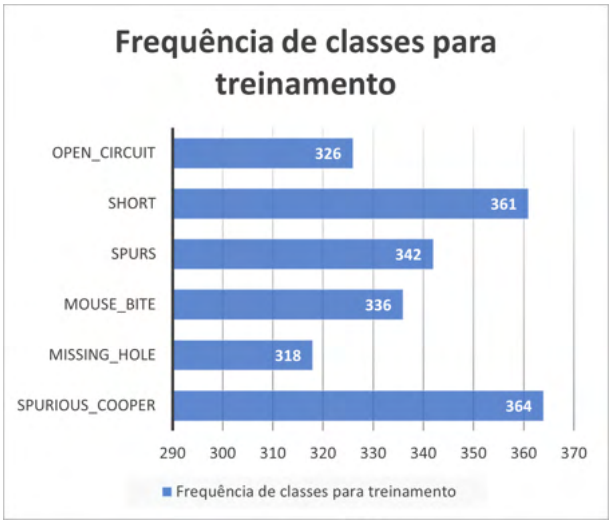


Figura 3.17: Frequência de classes para treinamento.

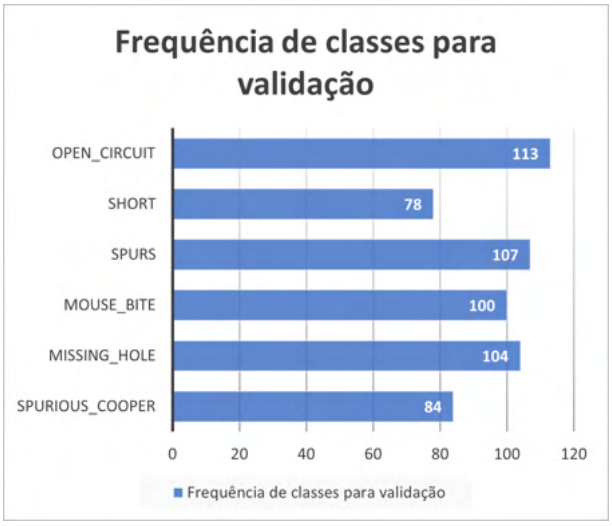


Figura 3.18: Frequência de classes para validação.

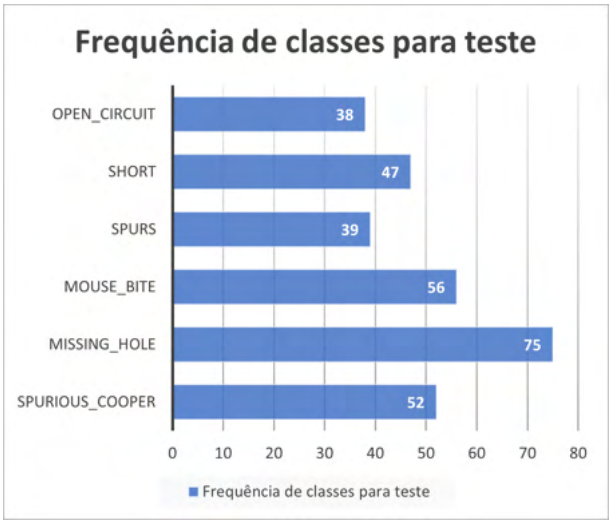


Figura 3.19: Frequência de classes para teste.

3.4.3 Execução

Após a divisão, os arquivos foram compactados em um único arquivo e importados para o Google Drive para serem utilizados na plataforma Google Colab.

Já na plataforma Google Colab, um novo arquivo de notebook foi criado para realização do experimento.

Os arquivos foram descompactados e o conjunto de programas yolov5 foram clonados do repositório ultralytics, commit sha 493981c. As dependências do programa foram instaladas conforme a especificação do arquivo requirements.txt.

As definições de localização de arquivos, número e nome de classes do *dataset* foram realizadas no arquivo dataset.yaml.

O notebook foi configurado para o uso de *Central Processing Unit* (CPU) Intel(R) Xeon(R) 2.20GHz e acelerador de hardware *Graphics Processing Unit* (GPU) Tesla T4.

Com a conclusão da clonagem do repositório e configuração do *dataset* foi iniciado o programa de treinamento, train.py, com base no modelo de rede Yolov5s, 300 épocas e batch size 16. O treinamento foi finalizado após 55 minutos de execução.

O programa de validação, val.py, foi executado e o resultado descrito na Tabela 3.2 foi obtido para *threshold*, limite de confiança, de 0,5. Durante a validação, as detecções com uma pontuação de confiança acima desse *threshold* são consideradas como positivas, enquanto aquelas abaixo são consideradas negativas.

Resultado para o conjunto de validação				
Classe	Instâncias	*Precision* (P)	*Recall* (R)	mAP
Todas as classes	586	0.97	0.903	0.949
missing_hole	104	0.998	1	0.995
mouse_bite	100	0.965	0.85	0.905
open_circuit	113	0.961	0.869	0.951
short	78	0.984	0.962	0.974
spur	107	0.964	0.785	0.901
spurious_cooper	84	0.947	0.952	0.971

Tabela 3.2: Resultados da validação.

Os resultados relacionados a *Precision*, P, dizem respeito a proporção de instâncias positivas corretamente previstas em relação ao total de instâncias previstas como positivas (verdadeiros positivos, TP, e falsos positivos, FP). A fórmula da precisão é dada por:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.1}$$

A precisão mede a qualidade das predições positivas do modelo, indicando quantas das instâncias previstas como positivas são verdadeiramente positivas.

Enquanto os resultados relacionados a *Recall*, R, são a a proporção de instâncias positivas corretamente previstas em relação ao total de instâncias que são verdadeiramente positivas (TP e falsos negativos, FN). A fórmula da revocação é dada por:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.2}$$

As previsões e resultados reais da validação resultaram na Figura 3.20, matrix de confusão.

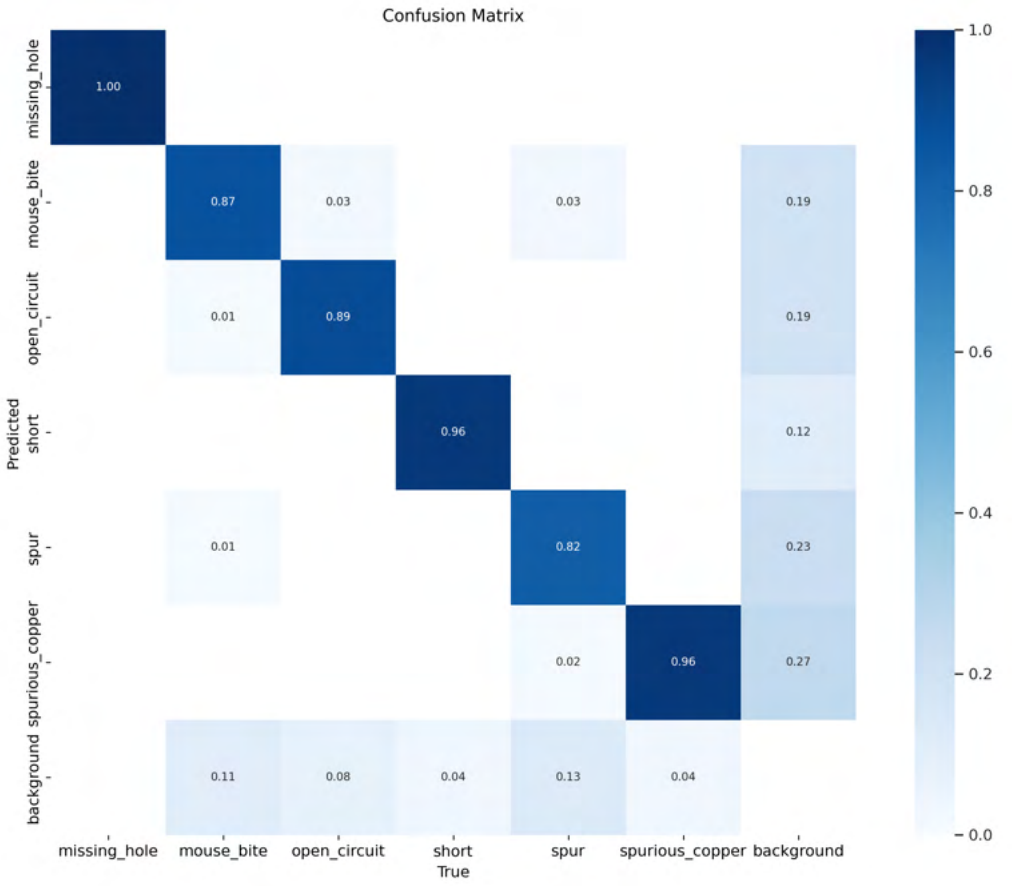


Figura 3.20: Matrix de Confusão da Validação.

O *Mean Average Precision* (mAP) indica a precisão média para todas as classes quando o threshold de confiança é definido como 0.5. Em resumo, o mAP de 0.949, obtido ao avaliar todas as classes, sugere que, em média, o modelo está alcançando uma precisão de 94,9% para detecções de objetos em todas as classes quando utiliza um threshold de confiança de 0.5. Este é um indicador geral da qualidade do modelo para a tarefa de detecção de objetos. Quanto maior o mAP, melhor é o desempenho do modelo.

A seguir, recortes de exemplos de detecção retornados pela execução da aplicação val.py. Obtidos durante a validação, Figura 3.21 até Figura 3.26. Nos recortes é possível visualizar as

detecções, através de caixas delimitadoras, classificações e nível de confiança, valor numérico expresso após o nome da classe.

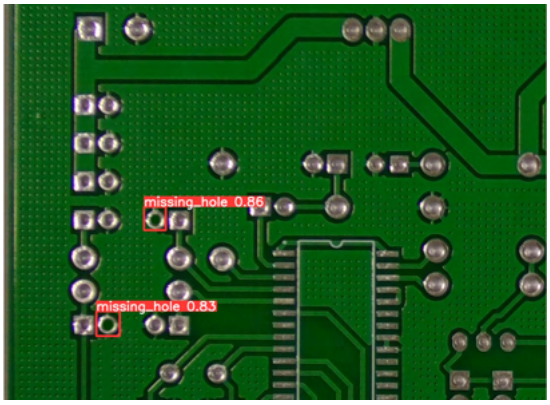


Figura 3.21: Detecção de missing hole.

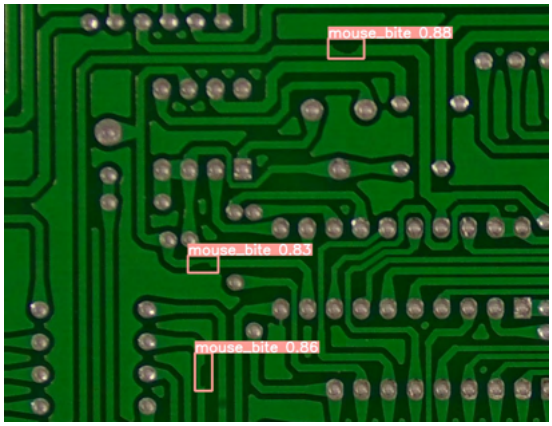


Figura 3.22: Detecção de mouse bite.

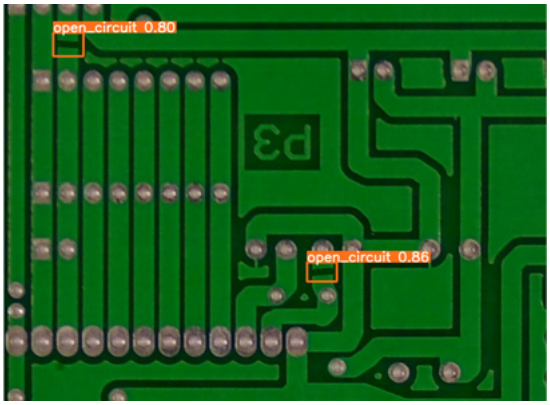


Figura 3.23: Detecção de open circuit.

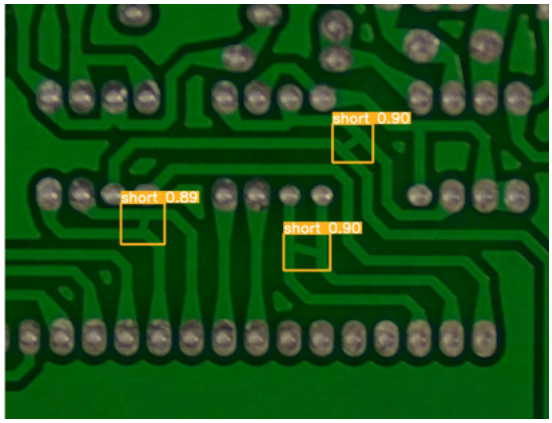


Figura 3.24: Detecção de short.

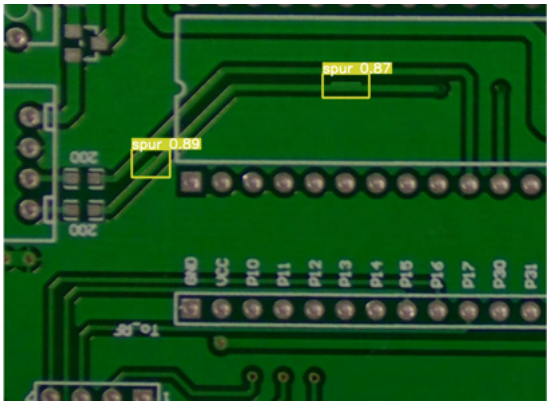


Figura 3.25: Detecção de spur.

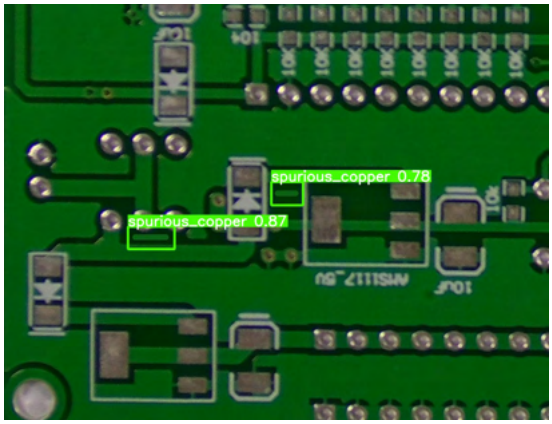


Figura 3.26: Detecção de spurious cooper.

Com o modelo treinado é aplicada a prova de conceito.

3.5 Prova de Conceito

O *dataset* obtidos na proposta de *software*, Figura 3.14, é importado para o *Google Drive*. Na plataforma *Google Colab* a aplicação detect.py é executada, com argumento "–save-txt" para gerar as anotações, e com os pesos obtidos na etapa de treinamento do modelo, Seção 3.4, referenciando os arquivos do *dataset* sintético como fonte de dados.

O resultado da execução permite concluir a detecção dos defeitos. A Figura 3.27 gerada após a detecção contém todas as imagens geradas pelo *software* proposto, conforme Figura 3.14, as caixas delimitadoras indicando a detecção, as classificações e seus respectivos níveis de confiança.

Portanto esta prova de conceito conclui que a proposta de *software* para geração de *datasets* de fato é capaz de gerar placas com defeitos aptas a serem utilizadas nas atividades de visão computacional para detecção de defeitos em PCBs, desde o treinamento até a detecção.

Nesta prova de conceito o menor nível de confiança foi de 80, o nível de confiança aceitável pode ser definido na aplicação detect.py conforme a necessidade via argumento "–conf ".

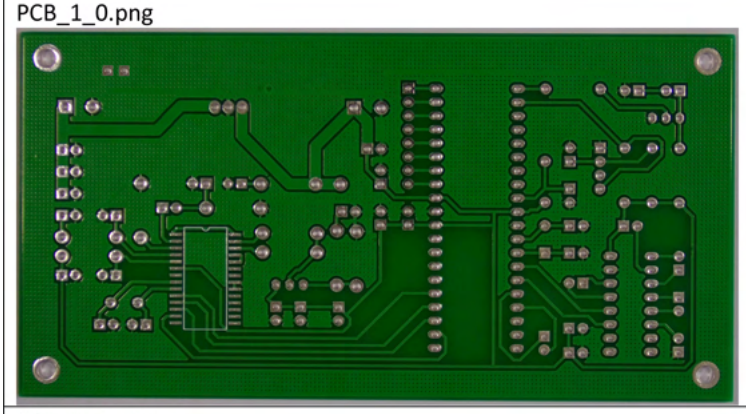
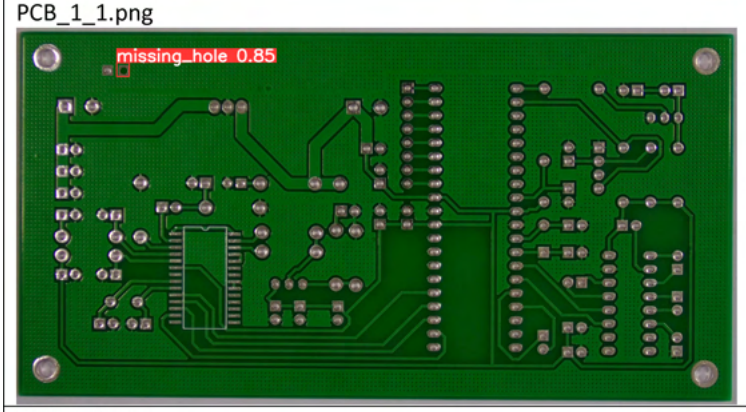
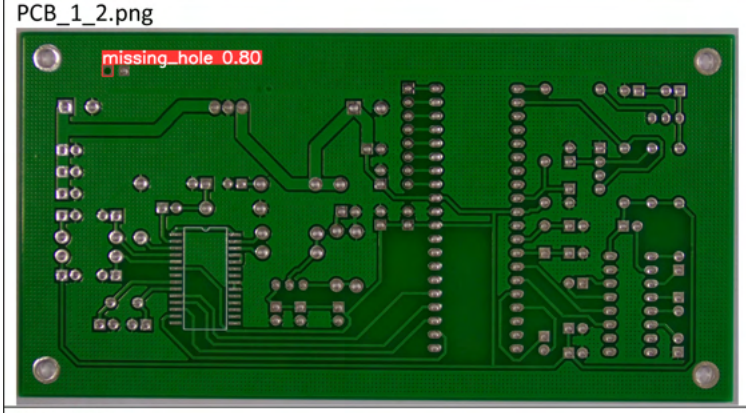
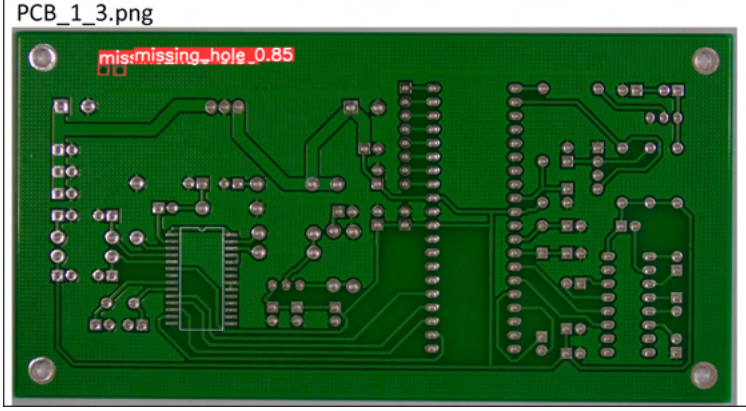
Imagens	Anotações
PCB_1_0.png 	Nenhum arquivo foi gerado
PCB_1_1.png 	PCB_1_1.txt: 0 0.147825 0.110971 0.0174687 0.03657
PCB_1_2.png 	PCB_1_2.txt: 0 0.126236 0.111917 0.0151615 0.0308953
PCB_1_3.png 	PCB_1_3.txt: 0 0.126236 0.110971 0.0145023 0.0290038 0 0.148484 0.110025 0.0168095 0.0334174

Figura 3.27: Prova de conceito.

Capítulo 4

Conclusões

O volume de desafios da inspeção óptica automática na detecção de defeitos de placas de circuito impresso resulta na necessidade de desenvolver sistemas AOI capazes de suprir as necessidades da alta demanda de fabricação de PCIs com qualidade. Sistemas com RNA de visão computacional, vêm sendo utilizados e apontados em estudos (WANG, C. et al., 2023a; SCHWEBIG, A.; TUTSCH, 2020; DENG; LUO; DAI, 2018). A escassez de dados públicos de treinamento para este tipo de solução é um fator limitante para o treinamento de RNA pois o desempenho de detecção de modelos de aprendizagem profunda é significativamente afetado devido à falta de amostras (WANG, C. et al., 2023a; SCHWEBIG, A.; TUTSCH, 2020). Na prática os conjuntos de dados são obtidos na produção industrial (WANG, C. et al., 2023a).

Como alternativa a este cenário, está dissertação propõe a criação de um software capaz de gerar conjuntos de dados sintéticos que permitam a identificação dos defeitos levantados na literatura. A partir desta proposta *datasets* podem ser criados e ampliados de modo a suprir as necessidades de treinado de RNA diversas, pois o *dataset* não é limitado a um RNA específica. Outras abordagens podem ser tratadas como por exemplo a criação de *datasets* para modelos distintos conforme a posição das estações de AOI em uma linha SMT, voltados a defeitos pertinentes ao momento no qual a análise é realizada. A prova de conceito, mesmo que enxuta, permite concluir a aplicabilidade da proposta.

A identificação e prevenção de defeitos durante a fabricação PCIs para EEE são extremamente importantes, em especial para casos em que um defeito de PCI possam resultar em danos e mortes, por exemplo na área de eletrônica automotiva, EEE de supervisão e controle etc (WANG, C. et al., 2023a).

Por fim, quanto aos trabalhos futuros, o desenvolvimento de novas funcionalidade de aumento de dados, criação de anotações em diferentes formatos, recursos de rotação e redimensionamento de imagens, integrações com softwares de desenvolvimento de PCIs, desenvolvimento de *datasets* colaborativos e distribuição são favoráveis a solução da escassez de dados de defeitos, o que contribui para o desenvolvimento de aplicações AOI.

Referências bibliográficas

AKHATOVA, A. (Ed.). **PCB Defects**. 2021. Disponível em: <https://www.kaggle.com/datasets/akhatova/pcb-defects>. Acesso em: 24 set. 2023.

ALIJAGIĆ, J.; ŠAJN, R. Predicting the Spatial Distributions of Elements in Former Military Operation Area Using Linear and Nonlinear Methods Across the Stavnja Valley, Bosnia and Herzegovina. **Minerals**, 2020. DOI: 10.3390/min10020120.

ASGHAR, R.; REHMAN, F.; AMAN, A.; IQBAL, K. Low RH and temperature effect on 0201 sized passive components during SMT mounting. **Soldering Surface Mount Technology**, Emerald, v. 32, n. 1, p. 48–54, ago. 2019. DOI: 10.1108/ssmt-02-2019-0006. Disponível em: <https://doi.org/10.1108/ssmt-02-2019-0006>.

BAGHERI, A. et al. A 16 × 16 45° Slant-Polarized Gapwaveguide Phased Array With 65-dBm EIRP at 28 GHz. **IEEE Transactions on Antennas and Propagation**, Institute of Electrical e Electronics Engineers (IEEE), v. 71, n. 2, p. 1319–1329, fev. 2023. DOI: 10.1109/tap.2022.3227718. Disponível em: <https://doi.org/10.1109/tap.2022.3227718>.

BUENO, A. C. et al. Analysis of Defects Generated by the Reflow Soldering in SMT (Surface Mount Technology) Assembly Applying the Six Sigma Method. **Journal of Integrated Circuits and Systems**, Journal of Integrated Circuits e Systems, v. 1, n. 2, p. 17–25, nov. 2020. DOI: 10.29292/jics.v1i2.259. Disponível em: <https://doi.org/10.29292/jics.v1i2.259>.

CHAUDHARY, V.; DAVE, I. R.; UPLA, K. P. Automatic visual inspection of printed circuit board for defect detection and classification. IEEE, mar. 2017. DOI: 10.1109/wispnet.2017.8299858. Disponível em: <https://doi.org/10.1109/wispnet.2017.8299858>.

CHOI, H. et al. Modulated dark-field phasing detection for automatic optical inspection. **Optical Engineering**, SPIE, v. 58, n. 9, p. 092603, 2019. DOI: 10.1117/1.OE.58.9.092603. Disponível em: <https://doi.org/10.1117/1.OE.58.9.092603>.

COMBET, C.; CHANG, M.-m. 01005 Assembly, the AOI route to optimizing yield, 2009. Disponível em: <https://api.semanticscholar.org/CorpusID:209852562>.

COSTA G. B. P.; PONTI, M. A. **Tópicos in Gerenciamento de Dados e Informações**. [S.l.]: MIT Press, 2017. cap. 3. Disponível em: <https://arxiv.org/pdf/1806.07908.pdf>.

DENG, Y.; LUO, A.; DAI, M. Building an automatic defect verification system using deep neural network for pcb defect classification, 2018. DOI: 10.1109/icfsp.2018.8552045.

DEY, D.; KUMAR, M. P. Yield Improvement in Wave Soldering Process by Using Customised Pallets. **Journal of University of Shanghai for Science and Technology**, ADD Technologies, v. 23, n. 06, p. 1001–1010, 2021. DOI: 10.51201/jusst/21/05343. Disponível em: <https://doi.org/10.51201/jusst/21/05343>.

DING, W. et al. How Do Open Source Communities Document Software Architecture: An Exploratory Survey. In: 2014 19th International Conference on Engineering of Complex Computer Systems. [S.l.]: IEEE, ago. 2014. DOI: 10.1109/iceccs.2014.26. Disponível em: <http://dx.doi.org/10.1109/iceccs.2014.26>.

FOWLER, M. **UML Essencial**. [S.l.: s.n.], 2006. cap. 9.

FUNG, V. W.; YUNG, K. C. An intelligent approach for improving printed circuit board assembly process performance in smart manufacturing. **International Journal of Engineering Business Management**, SAGE Publications, v. 12, p. 184797902094618, jan. 2020. DOI: 10.1177/1847979020946189. Disponível em: <https://doi.org/10.1177/1847979020946189>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. cap. 9. http://www.deeplearningbook.org.

HAQ, M. A.; JILANI, A. K.; PRABU, P. Algorithmic Scheme for Concurrent Detection and Classification of Printed Circuit Board Defects. **Computers, Materials Continua**, Computers, Materials e Continua (Tech Science Press), v. 71, n. 1, p. 355–367, 2022. DOI: 10.32604/cmc.2022.017698. Disponível em: <https://doi.org/10.32604/cmc.2022.017698>.

HAYKIN, S. **Redes Neurais - 2ed**. [S.l.: s.n.], 2001. P. 27, 32–36, 59.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep Residual Learning for Image Recognition, 2016. DOI: 10.1109/cvpr.2016.90.

HUANG, W.; WEI, P. A PCB Dataset for Defects Detection and Classification. arXiv, 2019. DOI: 10.48550/ARXIV.1901.08204. Disponível em: <https://arxiv.org/abs/1901.08204>.

KAUL, V.; ENSLIN, S.; GROSS, S. A. History of Artificial Intelligence in Medicine. **Gastrointestinal Endoscopy**, 2020. DOI: 10.1016/j.gie.2020.06.040.

KAYA, V.; AKGÜL, İ. Makine Öğrenmesi ve Derin Öğrenme Algoritmaları ile Baskı Devre Kartlarındaki Kusurların Tespiti. **European Journal of Science and Technology**, European Journal of Science e Technology, out. 2022. DOI: 10.31590/ejosat.1178188. Disponível em: <https://doi.org/10.31590/ejosat.1178188>.

KIM, D. et al. Rapid fault cause identification in surface mount technology processes based on factory-wide data analysis. **International Journal of Distributed Sensor Networks**, v. 15, n. 2, p. 1550147719832802, 2019. DOI: 10.1177/1550147719832802. eprint: https://doi.org/10.1177/1550147719832802. Disponível em: <https://doi.org/10.1177/1550147719832802>.

KÜHL, N.; GOUTIER, M.; HIRT, R.; SATZGER, G. Machine learning in artificial intelligence: towards a common understanding. **Proceedings of the Annual Hawaii International Conference on System Sciences**, 2019. DOI: 10.24251/hicss.2019.630.

KULIKOWSKI, C. A. BREVE INTRODUÇÃO À HISTÓRIA DA INTELIGÊNCIA ARTIFICIAL. **Jamaxi**, 2020. DOI: 10.1055/s-0040-1701972. Disponível em: <https://periodicos.ufac.br/index.php/jamaxi/article/view/4730>.

KULIKOWSKI, C. A. Donald A. B. Lindberg: Inspiring Leader and Visionary in Biomedicine, Healthcare, and Informatics. **Yearbook of Medical Informatics**, 2020. DOI: 10.1055/s-0040-1701972.

LAMBA, T.; INDIA, J.; MISHRA, A. Optimal machine learning model for software defect prediction. **International Journal of Intelligent Systems and Applications**, v. 11, p. 36–48, 2 2019. DOI: 10.5815/ijisa.2019.02.05.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, Institute of Electrical e Electronics Engineers (IEEE), v. 86, n. 11, p. 2278–2324, 1998. DOI: 10.1109/5.726791. Disponível em: <https://doi.org/10.1109/5.726791>.

LEGG, S.; HUTTER, M. Universal Intelligence: A Definition of Machine Intelligence. **Minds and Machines**, 2007. DOI: 10.1007/s11023-007-9079-x.

LIU, J. et al. Future paper based printed circuit boards for green electronics: fabrication and life cycle assessment. **Energy Environ. Sci.**, Royal Society of Chemistry (RSC), v. 7, n. 11, p. 3674–3682, 2014. DOI: 10.1039/c4ee01995d. Disponível em: <https://doi.org/10.1039/c4ee01995d>.

LIU, N.; SHAPIRA, P.; YUE, X. Tracking Developments in Artificial Intelligence Research: Constructing and Applying a New Search Strategy. **Scientometrics**, 2021. DOI: 10.1007/s11192-021-03868-4.

MAR, N.; YARLAGADDA, P.; FOOKES, C. Design and development of automatic visual inspection system for PCB manufacturing. **Robotics and Computer-Integrated Manufacturing**, v. 27, n. 5, p. 949–962, 2011. ISSN 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2011.03.007. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0736584511000457>.

MARTINEZ, I.; PEIRO-SIGNES, A. A review of the use of machine learning techniques in eco-innovation research, 2022. DOI: 10.4995/bmt2022.2022.15550.

MISTRY, M. et al. Parameter optimization for surface mounter using a self-alignment prediction model. **Soldering Surface Mount Technology**, Emerald, v. 35, n. 2, p. 78–85, jul. 2022. DOI: 10.1108/ssmt-01-2022-0008. Disponível em: <https://doi.org/10.1108/ssmt-01-2022-0008>.

MONETT, D. et al. Special Issue "On Defining Artificial Intelligence"—Commentaries and Author's Response. **Journal of Artificial General Intelligence**, 2020. DOI: 10.2478/jagi-2020-0003.

MUSTIKA, N.; NENDA, B.; RAMADHAN, D. Machine learning algorithms in fraud detection: case study on retail consumer financing company. **Asia Pacific Fraud Journal**, v. 6, p. 213, 2 2021. DOI: 10.21532/apfjournal.v6i2.216.

NGUYEN, V. N.; JENSSEN, R.; ROVERSO, D. Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning. **International Journal of Electrical Power & Energy Systems**, v. 99, p. 107–120, 2018. ISSN 0142-0615. DOI: https://doi.org/10.1016/j.ijepes.2017.12.016. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0142061517324444>.

NIELSEN, M. A. (Ed.). **Neural networks and deep learning**. 2022. Disponível em: <http://neuralnetworksanddeeplearning.com/about.html>. Acesso em: 22 nov. 2022.

PERDIGONES, F.; QUERO, J. Printed Circuit Boards: The Layers' Functions for Electronic and Biomedical Engineering. **Micromachines**, MDPI AG, v. 13, n. 3, p. 460, mar. 2022. DOI: 10.3390/mi13030460. Disponível em: <https://doi.org/10.3390/mi13030460>.

PRAMERDORFER, C.; KAMPEL, M. A dataset for computer-vision-based PCB analysis. IEEE, mai. 2015. DOI: 10.1109/mva.2015.7153209. Disponível em: <https://doi.org/10.1109/mva.2015.7153209>.

VU-QUOC, L.; HUMER, A. Deep Learning Applied to Computational Mechanics: A Comprehensive Review, State of the Art, and the Classics. **Computer Modeling in Engineering & Sciences**, 2023. DOI: 10.32604/cmes.2023.028130.

RAMANATHAN; AKASH; PRASAD, A.; RAHUL, V. Pick and Place Robot for Surface Mounting Devices. **International Research Journal on Advanced Science Hub**, RSP Science Hub, v. 2, n. 8, p. 75–81, set. 2020.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You only look once: unified, real-time object detection. **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, 2016. DOI: 10.1109/cvpr.2016.91.

REDMON, J.; FARHADI, A. Yolov3: an incremental improvement, 2018. DOI: 10.48550/arxiv.1804.02767.

RIZVI, A.; HALEEM, A.; BAHL, S.; JAVAID, M. **Artificial Intelligence (AI) and Its Applications in Indian Manufacturing: A Review**. [S.l.: s.n.], 2021. P. 825–835. DOI: 10.1007/978-981-33-4795-3_76.

ROBOFLOW (Ed.). **RoboFlow: Give yout software the power to see objects in images and video**. 2023. Disponível em: <https://roboflow.com/>. Acesso em: 10 ago. 2023.

RODRIGUES, L.; ALBUQUERQUE, M.; PEREIRA, A.; AZZI, G. Fabricação de Placa de Circuito Impresso (PCI) usando Método Fotográfico e Redução Significativa de Soluções Químicas. **Notas Técnicas**, Brazilian Center for Physical Research, v. 9, n. 3, p. 26–33, abr. 2019. DOI: 10.7437/nt2236-7640/2019.03.009. Disponível em: <https://doi.org/10.7437/nt2236-7640/2019.03.009>.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, v. 65, n. 6, p. 386–408, 1958. ISSN 0033295X. DOI: 10.1037/h0042519.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence, A Modern Approach**. [S.l.: s.n.], 2011. P. 3–7. ISBN 0131038052.

SCHWEBIG, A.; TUTSCH, R. Compilation of training *datasets* for use of convolutional neural networks supporting automatic inspection processes in industry 4.0 based electronic manufacturing. **Journal of Sensors and Sensor Systems**, v. 9, p. 167–178, 1 2020. DOI: 10.5194/jsss-9-167-2020.

SCHWEBIG, A. I. M.; TUTSCH, R. Intelligent fault detection of electrical assemblies using hierarchical convolutional networks for supporting automatic optical inspection systems. **Journal of Sensors and Sensor Systems**, v. 9, n. 2, p. 363–374, 2020. DOI: 10.5194/jsss-9-363-2020. Disponível em: <https://jsss.copernicus.org/articles/9/363/2020/>.

SELBMANN, F. et al. Paradigm Changing Integration Technology for the Production of Flexible Electronics by Transferring Structures, Dies and Electrical Components from Rigid to Flexible Substrates. **Micromachines**, MDPI AG, v. 14, n. 2, p. 415, fev. 2023. DOI: 10.3390/mi14020415. Disponível em: <https://doi.org/10.3390/mi14020415>.

SHAOBIN, Y. et al. An optical apparatus for inspecting adjacent surfaces defects of TEC components with equal-optical-path confocal imaging using optical wedge prisms. **Journal of Physics: Conference Series**, IOP Publishing, v. 2226, n. 1, p. 012005, 2022. DOI: 10.1088/1742-6596/2226/1/012005. Disponível em: <https://dx.doi.org/10.1088/1742-6596/2226/1/012005>.

TACCHINO, F.; MACCHIAVELLO, C.; GERACE, D.; BAJONI, D. An Artificial Neuron Implemented on an Actual Quantum Processor. **NPJ Quantum Information**, 2019. DOI: 10.1038/s41534-019-0140-4.

TAN, L.; HUANGFU, T.; WU, L.; CHEN, W. Comparison of yolo v3, faster r-cnn, and ssd for real-time pill identification, 2021. DOI: 10.21203/rs.3.rs-668895/v1.

WANG, C.; HUANG, G.; HUANG, Z.; HE, W. Conditional transgan-based data augmentation for pcb electronic component inspection. **Computational Intelligence and Neuroscience**, v. 2023, p. 1–12, 2023. DOI: 10.1155/2023/2024237.

WANG, C.; HUANG, G.; HUANG, Z.; HE, W. Conditional TransGAN-based data augmentation for PCB electronic component inspection. en. v. 2023, p. 2024237, jan. 2023.

WESTNY, T. Data-Driven Interaction-Aware Behavior Prediction for Autonomous Vehicles, 2023. DOI: 10.3384/9789180751797.

WU, W.-Y.; WANG, M.-J. J.; LIU, C.-M. Automated inspection of printed circuit boards through machine vision. **Computers in industry**, Elsevier, v. 28, n. 2, p. 103–111, 1996.

YAMANE, L. H.; ESPINOSA, D. C. R.; TENÓRIO, J. A. S. LIXIVIAÇÃO BACTERIANA DE SUCATA ELETRÔNICA: INFLUÊNCIA DOS PARÂMETROS DE PROCESSO. **Tecnologia em Metalurgia Materiais e Mineração**, Editora Cubo, v. 10, n. 1, p. 50–56, 2013. DOI: 10.4322/tmm.2013.007. Disponível em: <https://doi.org/10.4322/tmm.2013.007>.

YAMASHITA, R.; NISHIO, M.; GIAN, R. K.; TOGASHI, K. **Convolutional Neural Networks: An Overview and Application in Radiology**. [S.l.: s.n.], 2018. DOI: 10.1007/s13244-018-0639-9.

YU, X.; HE, Y. PCB defect detection based on GAN data generation with self-attentive mechanism. In: 2022 2nd International Conference on Frontiers of Electronics, Information and Computation Technologies. Wuhan, China: IEEE, ago. 2022.

YU, X.; SHU, C.-W. Multi-Layer Perceptron Estimator for the Total Variation Bounded Constant in Limiters for Discontinuous Galerkin Methods. **La Matematica**, 2021. DOI: 10.1007/s44007-021-00004-9.

ZHAO, H.; CHENG, J.; JIN, J. NI vision based automatic optical inspection (AOI) for surface mount devices: Devices and method. In: 2009 International Conference on Applied Superconductivity and Electromagnetic Devices. [S.l.: s.n.], 2009. P. 356–360. DOI: 10.1109/ASEMD.2009.5306622.