UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE TECNOLOGIA

MARCOS ANTONIO ANDRADE

Identificação de anomalias em placas de circuito impresso, propostas de registro e classificação.

Identification of anomalies in printed circuit boards, registration and classification proposals.

Limeira
2022

MARCOS ANTONIO ANDRADE

Identificação de anomalias em placas de circuito impresso, propostas de registro e classificação.

Identification of anomalies in printed circuit boards, registration and classification proposals.

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

ORIENTADOR: PROF. DR. LEANDRO RONCHINI XIMENES
COORIENTADOR: PROF DR. RANGEL ARTHUR

Este trabalho corresponde à versão final da Dissertação defendida por Marcos Antonio Andrade e orientada pelo Prof. Dr. Leandro Ronchini Ximenes

Limeira
2022

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

An24i

Andrade, Marcos Antonio, 1978-
Identificação de anomalias em placas de circuito impresso, propostas de registro e classifcação / Marcos Antonio Andrade. – Limeira, SP : [s.n.], 2022.

Orientador: Leandro Ronchini Ximenes.
Coorientador: Rangel Arthur.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Tecnologia.

1. Segmentação de imagens. 2. Transformada watershed. I. Ximenes, Leandro Ronchini, 1985-. II. Arthur, Rangel, 1977-. III. Universidade Estadual de Campinas. Faculdade de Tecnologia. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Identification of anomalies in printed circuit boards, registration and classification proposals
Palavras-chave em inglês:
Image segmentation
Watershed transform
Área de concentração: Sistemas de Informação e Comunicação
Titulação: Mestre em Tecnologia
Banca examinadora:
Leandro Ronchini Ximenes [Orientador]
Edson Luiz Ursini
Alexandre Gonçalves Silva
Data de defesa: 07-04-2022
Programa de Pós-Graduação: Tecnologia

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-9384-1129
- Currículo Lattes do autor: http://lattes.cnpq.br/5311669528806297

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 Marcos Antonio Andrade, em 07 de Abril de 2022 na Faculdade de Tecnologia FT/UNICAMP, em Limeira/SP.

Prof. Dr. Leandro Ronchini Ximenes
Presidente da Comissão Julgadora

Prof. Dr. Edson Luiz Ursini
Universidade Estadual de Campinas.

Prof. Dr. Alexandre Gonçalves Silva
Universidade Federal de Santa Catarina.

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.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Resumo

Com a crescente concorrência entre os diversos fabricantes de produtos eletrônicos, a qualidade dos produtos desenvolvidos e a consequente confiança na marca são fatores fundamentais na sobrevivência das empresas. Um dos pilares para garantir a qualidade dos produtos na manufatura é a inspeção visual, sendo o objetivo do trabalho estudar os diversos métodos para a detecção de anomalias e propor um comitê, composto por métodos tradicionais e de aprendizado de máquina. Este comitê integra um sistema automático de inspeção visual capaz de segregar as placas de circuito impresso que possuam algum tipo de anomalia. Experimentos com um determinado modelo de *smartphone* mostram um *IoU Score* de 98,46% no registro do *bracket*, enquanto que o comitê proposto alcançou uma acurácia média de aproximadamente 90,87% na detecção de defeitos, além de uma taxa de falso alarme de 5,32%, considerando 22 componentes por placa.

Palavras-chave: PCB, classificação de defeito, segmentação semântica, U-Net, *watershed*.

Abstract

With the growing competition between the various manufacturers of electronic products, the quality of the products developed and the consequent confidence in the brand are fundamental factors in the survival of companies. One of the pillars to guarantee the quality of the products in the manufacture is the visual inspection, being the objective of the work to study the several methods for the detection of anomalies and to propose a committee, composed by traditional methods and machine learning. This committee integrates an automatic visual inspection system capable of segregating printed circuit boards that have some type of anomaly. Experiments with a particular smartphone model show an IoU Score of 98.46% in the bracket registration, while the proposed ensemble achieved an average accuracy of approximately 90.87% in defect detection, in addition to a false alarm rate of 5.32%, considering 22 components each board.

Key words: PCB, defect classification, semantic segmentation, U-Net, *watershed*.

Índice de Figuras

Índice de Tabelas

Lista de Abreviaturas e Siglas

Siglas

2D DFT – *2D Discrete Fourier Transform*

3C – *Computer, Communication and Consumer Electronics*

Adam – *Adaptive Moment Estimation*

AOI – *Automated Optical Inspection*

AUC – *Area Under Curve*

AUROC - *Area Under ROC Curve*

BFMatcher - *Brute Force Matcher*

Blob – *Binary Large Objects*

BOM – *Bill Of Materials*

CAE – *Convolutional AutoEncoder*

CI – *Circuitos Integrados*

CNN – *Convolutional Neural Network*

DA – *Data Augmentation*

DCT – *Discrete Cosine Transform*

DL – *Deep Learning*

DoG – *Difference of Gaussian*

FAST – *Features from Accelerated Segment Test*

FCN – *Fully Convolutional Network*

FN - *False Negatives*

FP – *False Positives*

FPN – *Feature Pyramid Network*

FSL – *Few Samples Learning*

GAN – *Generative Adversarial Networks*

GFTT – *Good Features to Track*

GN – *Graph Network*

GPU – *Graphics Processing Unit*

HOG – *Histogram of Oriented Gradients*

IA – *Inteligência Artificial*

ICT – *In Circuit Test*

INCC – *Improved Normalized Cross-Correlation*

IOT – *Internet of Things*

IoU – *Intersection Over Union*

JSON - *JavaScript Object Notation*

LBP – *Local Binary Pattern*

lr – *learning rate*

M2M – *Machine to Machine*

mAP – *mean Average Precision*

ML – *Machine Learning*

MRHAM – *Multistage Residual Hybrid Attention Module*

MSAC – *M-estimator Sample Consensus*

MSE – *Mean Squared Error*

NCC – *Normalized Cross-Correlation*

NMS – *Non-Maximum Suppression*

NN – *Neural Networks*

ORB – *Oriented FAST and Rotated BRIEF*

PCB – *Printed Circuit Board*

RAM – *Random Access Memory*

RANSAC – *Random Sample Consensus*

RCNN – *Regional Based Convolutional Neural Network*

ReLU – *Rectified Linear Unit*

RMSE – *Root Mean Square Error*

RPN – *Region Proposal Network*

SGD – Stochastic Gradient Descent

SIFT – *Scale Invariant Feature Transform*

SMD – *Surface Mounted Device*

SPI – *Solder Past Machine*

SPN – *Similarity Prediction Network*

SSD – *Sum of Squared Difference*

SSDT – *Single Shot object DeTector*

SSIM – *Structural Similarity Index*

SSIM – *Structural Similarity Index Measure*

SURF – *Speeded-Up Robust Features*

SVM – *Support Vector Machine*

TL – *Transfer Learning*

TN – *True Negative*

TP – *True Positive*

UAD-GAN – *Unsupervised Anomaly Detection GAN*

UNICAMP – *Universidade Estadual de Campinas*

WGAN-GP – *Wasserstein GAN with Gradient Penalty*

WKDE – *Weighted Kernel Density Estimator*

YOLO – *You Only Look Once*

Sumário

1 Introdução

A produção de produtos eletrônicos vem aumentando nos últimos anos, principalmente devido ao aparecimento de uma nova gama de produtos: *smartphones*, televisores inteligentes (*Smart TVs*), relógios inteligentes (*Smartwatches)*, drones e robôs [1]. Mesmo no meio industrial, com o advento da Internet das Coisas (IoT – *Internet of Things*) e da comunicação máquina a máquina (*M2M – Machine to Machine*), os circuitos eletrônicos das máquinas de produção estão cada vez mais especializados, sendo capazes, por exemplo, de enviar informação do seu funcionamento para os sistemas supervisórios. Em quaisquer destes cenários, os produtos eletrônicos modernos apresentam circuitos cada vez mais complexos, com dimensões cada vez menores, dificultando assim o seu processo produtivo e dando margem a um maior aparecimento de erros.

Os equipamentos eletrônicos normalmente consistem de uma placa de circuito impresso (PCB – *Printed Circuit Board*) principal e diversas outras placas com circuitos auxiliares, além do acondicionamento mecânico. A PCB é composta por uma placa de material isolante, onde são impressas trilhas condutivas de acordo com o projeto do circuito, além de pontos específicos onde os componentes do circuito, como, capacitores, resistores, transistores, circuitos integrados, indutores, entre outros são soldados [2], [3].

A PCB é uma das partes mais sensíveis ao aumento de complexidade na produção dos novos produtos eletrônicos. Atualmente, devido ao crescente aumento da demanda de consumo destes produtos aliado ao surgimento de diversos fabricantes, transformou a garantia da qualidade do processo de produção de PCB em um fator determinante na competição por mercado [3] .

O processo de montagem dos equipamentos eletrônicos, inclusive a PCB, é composto por etapas de montagem, podendo estas estarem distribuídas em linha ou em células de produção. Em cada etapa os operadores e máquinas realizam um processo específico, seja a inserção de componentes, montagem mecânica ou testes de verificação. Devido a fatores como escassez de mão de obra especializada, de tecnologia disponível ou ausência/presença de incentivos fiscais regionais, muitas empresas dividem o processo produtivo entre as suas diversas plantas, sendo estas localizadas em regiões geográficas distantes. No transporte de

tais produtos semiacabados existe a possibilidade que eles sejam danificados, além disso, deve-se considerar também erros humanos, erros do maquinário envolvido e lotes de componentes defeituosos [4].

Muitos dos defeitos introduzidos no processo de montagem só podem ser observados no teste final do produto acabado. Tal teste é realizado pelo departamento de qualidade, onde os colaboradores simulam o uso do produto pelo usuário final. Os testes no departamento de qualidade são feitos de forma amostral (inspeção por amostragem para aceitação), quando um erro é encontrado todo lote é reprovado, em tese todos os produtos do lote são testados [5]. Os produtos reprovados são submetidos ao retrabalho, o que causa um aumento do custo produtivo. Ou, não raramente, no desmonte do produto as peças são danificadas, sendo necessário o descarte. Ademais, alguns produtos defeituosos, visto que o teste é amostral, acabam chegando ao consumidor final. Defeitos que são encontrados durante o uso pelo consumidor final acarretam a perda de confiança na marca [6]. Assim, a qualidade da montagem da PCB influencia na vida útil do produto o que resulta em uma maior confiança por parte do consumidor [7].

Para minimizar esse problema, as empresas de produtos eletrônicos utilizam testes elétricos (*ICT – In Circuit Test*) e inspeção visual dos componentes entre as diversas etapas da produção. Os testes elétricos geralmente são realizados por máquinas que excitam determinados circuitos do produto eletrônico, validando as suas funcionalidades. Deste modo, o projeto do circuito elétrico precisa contemplar pontos de contato externo possibilitando a injeção de corrente ou tensão de modo a excitar o circuito. Porém, muitas vezes não existe essa preocupação do projetista ou fabricante no momento do projeto, fazendo com que os testes elétricos não sejam capazes de avaliar todo o circuito [8], [9]. Por fim, mesmo as máquinas de teste podem causar defeitos nos componentes durante a movimentação das placas.

Já a inspeção visual é feita geralmente por meio de sistemas automáticos (AOI - *Automated Optical Inspection*), cujos complexos softwares de processamento de imagem são usados para identificar os diversos tipos de defeitos em PCB. As máquinas de AOI não são normalmente projetadas para um produto específico, devendo ser capazes de detectar o maior número de falhas nos diversos modelos de PCB existentes no mercado. Deste modo, o *software* de tais máquinas possuem uma grande quantidade de parâmetros, o que torna o uso destes *softwares* complexos por si só. Soma-se a isso a constante evolução dos produtos eletrônicos,

resultando numa maior demanda por profissionais altamente qualificados, com conhecimentos avançado em métodos de processamento de imagem.

Segundo [8] os sistemas AOI, considerando diversos fabricantes, conseguem identificar, em média, de 40% a 50% dos defeitos. Assim, mesmo com a existência de sistemas automáticos, boa parte das inspeções visuais, principalmente após as montagens de SMD (*Surface Mounted Device*), ainda são feitas pelos operadores a olho nu ou com a ajuda de algum microscópio. Porém, o julgamento do operador está relacionado com a sua experiência e capacidade de concentração. Um ponto agravante é que geralmente o operador necessita entrar com os dados do produto no sistema de chão de fábrica, além de realizar a inspeção ele mesmo.

Durante a jornada de trabalho, o operador vai perdendo a sua capacidade de concentração, principalmente devido à realização de outras atribuições profissionais e à diminuição do tempo de inspeção, ocasionada pelo aumento da produção em determinados momentos do turno. Observa-se também uma dificuldade de se encontrar mão de obra qualificada, pois as novas gerações de trabalhadores são relutantes em trabalhar em processos repetitivos que não agregam valor [1]. Em resumo, a inspeção visual utilizando operadores humanos é passível de inconsistências, com os resultados podendo variar de uma pessoa para outra, além de ser um processo demorado e tedioso [6], [10], [11].

É nesse contexto que o presente trabalho deseja atuar, propondo métodos de inspeção visual capazes de identificar anomalias em PCB, além de simplificar a parametrização exigida pelos softwares dos equipamentos de AOI. A substituição do operador irá remover o caráter subjetivo da inspeção, enquanto que a simplificação na parametrização irá facilitar o treinamento de novos profissionais e permitir uma maior velocidade na troca de configuração para novos modelos.

Além disso, os métodos serão executados em computadores e irão alimentar banco de dados de produção e máquinas automáticas de segregação de peças defeituosas. Por meio das informações obtidas e da automação, será possível prover dados estatísticos permitindo a melhoria dos processos de fornecedores, além de ocasionar um aumento na produção por meio da diminuição do *takt time*. O *takt time* mede o compasso ou ritmo de produção em sistemas de produção do tipo puxado [12]. Como a inspeção visual feita por humanos tende a ser mais demorada, a construção de um sistema de AOI irá acelerar a produção diminuindo o *takt time*.

Embora exista uma lista de itens a serem inspecionados, definida para cada modelo das PCB fabricadas, essa lista é revista de acordo com a maturação do processo produtivo. Em outras palavras, à medida que os modelos vão sendo produzidos, melhorias de processo, treinamento de mão de obra e melhorias nas metodologias de teste são aplicadas. Deste modo, os defeitos tornam-se menos frequentes, elementos que não apresentam defeito durante um determinado tempo deixam a lista de inspeção visual. Por outro lado, novos fornecedores de componentes ou mudanças de projeto podem surgir, fazendo com que novos componentes sejam inseridos na lista ou mesmo que outros retornem à inspeção.

Existem diversos trabalhos que propõem sistemas AOI por meio do uso de métodos de processamento de imagens tradicionais, tais como: registro de uma imagem em relação a uma PCB referência, correlação de componentes através da extração de características (*features*) e subtração de imagens. Porém, considerando que a inspeção a ser realizada é modificada a todo tempo e a velocidade de lançamento de novos produtos na indústria eletrônica, o sistema de AOI necessita ser facilmente adaptável à novas regras e produtos. Assim, é interessante ao sistema AOI apresentar características de inteligência artificial (IA).

Dentre os diversos campos da IA, o aprendizado de máquina (ML – *Machine Learning*) está recebendo uma grande atenção, principalmente devido aos recentes avanços, relacionados à visão computacional, alcançados por uma classe específica dentro das redes neurais artificiais: aprendizado profundo (DL – *Deep Learning*) [13]–[15]. Segundo [13], métodos de DL como as redes neurais convolucionais (*CNN – Convolutional Neural Network*) estão sendo utilizadas para resolver problemas complexos tais como: segmentação, classificação e detecção de imagens, apresentando inclusive, um desempenho melhor quando comparado aos métodos tradicionais. Esses problemas complexos apresentados estão intimamente relacionados com a inspeção visual proposta neste trabalho.

Ainda segundo [13], o DL introduz o conceito *end-to-end learning*, onde o modelo DL é treinado a partir de um conjunto de dados e a rede neural descobre os padrões necessários para classificação das imagens. Esse conceito encaixa-se perfeitamente no modelo de inspeção visual desejado à indústria eletrônica. Portanto, o presente trabalho irá apresentar um conjunto de CNNs capazes de segmentar a PCB de produtos eletrônicos, a partir de imagens destas sobre a esteira na linha de produção e encontrar anomalias de montagem. As anomalias a

serem avaliadas neste trabalho em geral, são representadas por componentes ausentes, posicionamento errado dos componentes e componentes amassados.

Porém ainda considerando o descrito em [13], a abordagem tradicional não pode ser descartada. Algumas classes de problemas apresentam melhores resultados quando métodos tradicionais são utilizados. Deste modo, o trabalho irá comparar o registro da PCB e a detecção de anomalias, utilizando os métodos tradicionais e as redes neurais. Os métodos mais eficientes serão organizados em um comitê a ser utilizado na avaliação final da montagem da PCB.

Embora os métodos a serem apresentados possam ser utilizados em qualquer indústria de fabricação eletrônica, os casos de teste serão avaliados utilizando-se imagens de PCB de *smartphones*. Na linha de produção de *smartphones* utilizada como referência, existem dois pontos onde a inspeção visual será relevante na garantia da qualidade, na entrada de produtos oriundos de outras plantas e no processo anterior à montagem final. Em relação ao primeiro processo serão apenas apresentadas algumas imagens, com o intuito de exemplificar a possibilidade de uso da inspeção. Os resultados e treinamentos serão feitos utilizando-se imagens do segundo processo citado.

No processo anterior à montagem final, a PCB principal do produto encontra-se acondicionada em um acabamento mecânico. Esse conjunto recebe o nome de *bracket*. Os seguintes componentes geralmente estão visíveis nesse momento: motor de vibração, antenas de contato, conectores, *flat cables*, componentes SMD, *ear jack* e pontos de aterramento.

A Figura 1 exemplifica os itens a serem inspecionados no processo de entrada de produtos semiacabados, além de exemplos de possíveis anomalias. Em (a) é apresentado um componente ausente que foi removido durante os testes, em (b) é apresentado um componente ausente que não foi inserido de maneira correta e em (c) tem-se um ponto de aterramento danificado.

A Figura 2 exemplifica uma inspeção visual a ser realizada antes da montagem final do aparelho, com os componentes a serem inspecionados apontados na imagem. Além disso, são mostrados 4 exemplos de anomalias em alguns componentes.

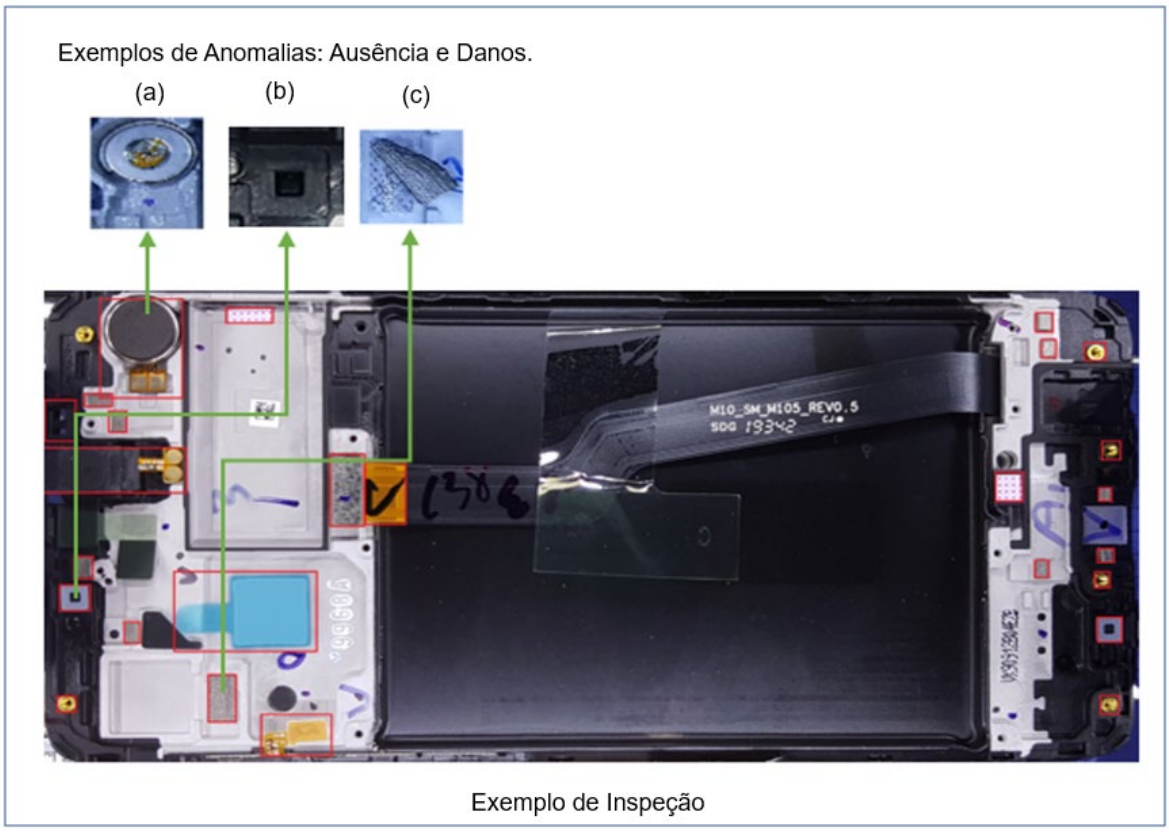


Figura 1 - Exemplos de itens inspecionados e anomalias. (a) e (b) componentes ausentes, (c) componente danificado.

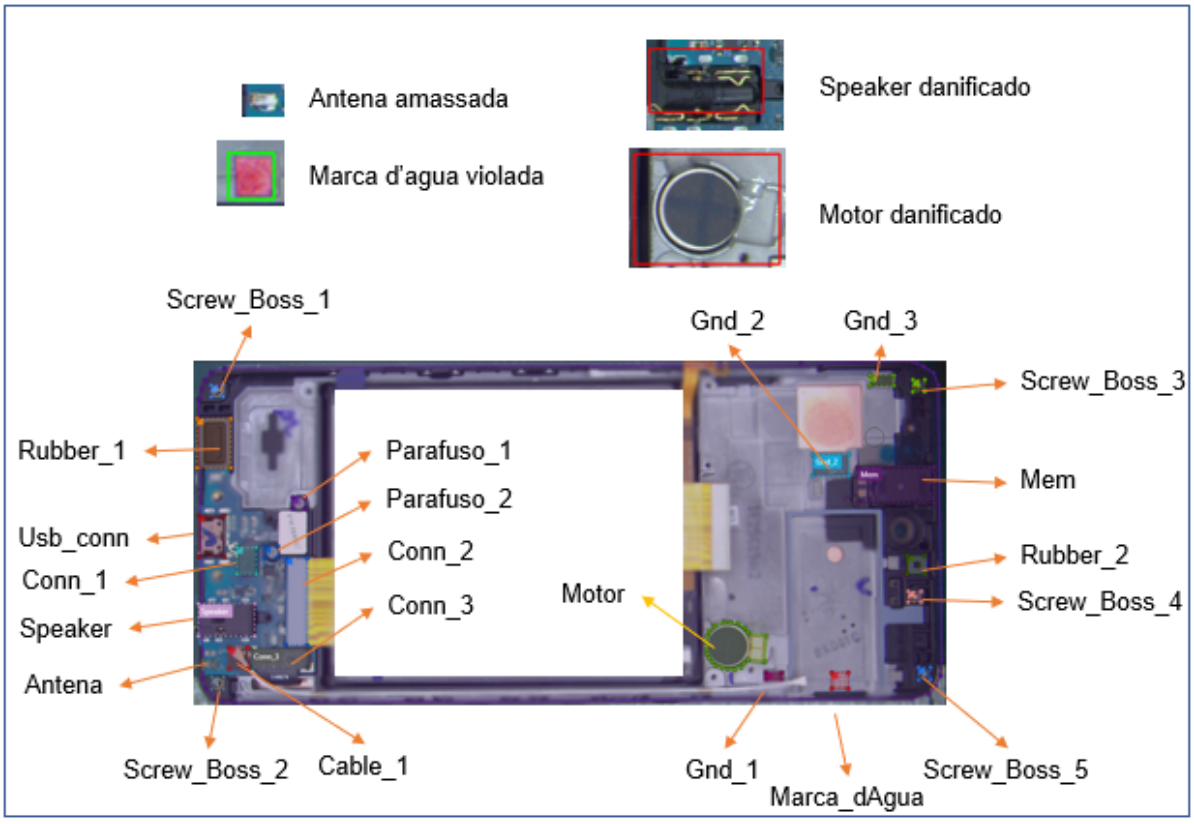


Figura 2 - Bracket, exemplo de inspeção e anomalias.

1.1 Objetivo do trabalho

O objetivo do projeto é encontrar um comitê de métodos, formado por CNNs e métodos tradicionais de processamento de imagem, capazes de registrar o *bracket* e identificar as anomalias, sendo que tais métodos serão treinados fazendo o uso somente de imagens sem defeitos.

As imagens dos *brackets* são adquiridas por meio de uma câmera Basler acA4096-11gc, que é uma câmera colorida com resolução de 9 mega pixels e acionamento *global shutter*. A câmera encontra-se posicionada sobre a esteira de transporte do produto, a iluminação é feita por um dome de luz branca e acoplada à câmera tem-se uma lente de 12 mm.

Quando a peça aciona o sensor de gatilho, a imagem é capturada pela câmera, sendo que a esteira não interrompe o seu movimento. Essa etapa do processo produtivo é localizada antes da montagem final do aparelho. Porém, os métodos propostos devem ser adaptáveis a outros processos e produtos, sendo necessário apenas um novo treinamento e definição de alguns parâmetros.

1.2 Contribuições

As principais contribuições a serem apresentadas pelo presente trabalho são:

- Proposta de uma base de treinamento baseada em amostras sem defeito;
- Proposta do hardware necessário para a aquisição e classificação das imagens em tempo real;
- Proposta de algoritmo de registro para PCB com acurácia média acima de 90,00%;
- Proposta de um método para classificação automática de componentes defeituosos em PCB com acurácia média acima de 90,00%;
- Dados comparativos entre métodos que não utilizam redes neurais e métodos de DL;
- Publicação do artigo “*A SURVEY ON AUTOMATIC INSPECTION FOR PRINTED CIRCUIT BOARD ANALYSIS*” no *7th Brazilian Technology*

Symposium'21. Este artigo também será publicado como capítulo de livro pela editora *Springer*.

1.3 Organização do Trabalho

O Capítulo 2 apresenta os fundamentos teóricos referentes os seguintes temas: aquisição e representação de imagens digitais, redes neurais, definição de funções de custo, definição de medidas de desempenho, *Autoencoders* e GAN (*Generative Adversarial Network)*. Além de métodos tradicionais de processamento de imagens, como por exemplo o *watershed,* transformada discreta de Fourier 2D e o algoritmo de classificação *K-means*.

No Capítulo 3 é apresentada uma revisão bibliográfica, onde são listados trabalhos que abordam métodos tradicionais e métodos baseados em IA (Inteligência Artificial) aplicados na análise de trilhas e componentes de PCB. Nesse capítulo, também é apresentado um resumo comparativo dos métodos citados. Este resumo mostra os algoritmos utilizados, as quantidades de imagens utilizadas e os resultados obtidos.

No Capítulo 4 é apresentada a metodologia utilizada, um pequeno conjunto de imagens demonstra, de forma didática, as transformações efetuadas pelos métodos propostos.

Os resultados do trabalho são apresentados no Capítulo 5, onde são discutidas as acurácias encontradas na avaliação de cada componente de um *bracket* específico. O cálculo do *IoU Score*, considerando o *bracket* e componentes de um segundo modelo de *smartphone*, finaliza esse capítulo.

No Capítulo 6, as conclusões e os benefícios da aplicação do método proposto na linha de produção de *smartphones*, são apresentados.

2 Fundamentação Teórica

Neste capítulo são apresentados os fundamentos teóricos referentes aos temas utilizados na implementação do comitê de métodos proposto no trabalho. Inicialmente, são discutidas as tecnologias de câmeras utilizadas na aquisição das imagens e a representação dessas imagens no mundo digital. Posteriormente, são apresentados os conceitos de redes neurais e métodos clássicos de processamento de imagens.

2.1 Aquisição de Imagens

Nas antigas câmeras que utilizavam filmes, o obturador controlava a quantidade de luz a entrar na câmera e, consequentemente, influenciava a qualidade da foto. Atualmente, os filmes foram substituídos por sensores, porém o controle do tempo de exposição à luz proporcionado pelo obturador ainda é um conceito importante na aquisição de imagens. Existem dois métodos de exposição dos sensores à luz: *Global Shutter* e *Rolling Shutter* [16], [17].

Nas câmeras com suporte ao método *Global Shutter*, quando o obturador abre todos os sensores capturam a luz ao mesmo tempo fazendo com que a imagem seja capturada de uma única vez. Esse método é recomendado para capturar imagens de objetos em movimento. Já nas câmeras *Rolling Shutter*, quanto o obturador abre, as linhas de sensores são expostas sucessivamente, linha após linha. Considerando a construção eletrônica deste tipo de câmera, ela é menos susceptível ao aquecimento e possui um menor tamanho quando comparada as câmeras *Global Shutter*, porém pode causar distorção ao capturar objetos em movimento. A Figura 3

e a Figura 4 exemplificam a aquisição da imagem de um objeto em movimento utilizando os dois tipos de câmeras [16], [18].

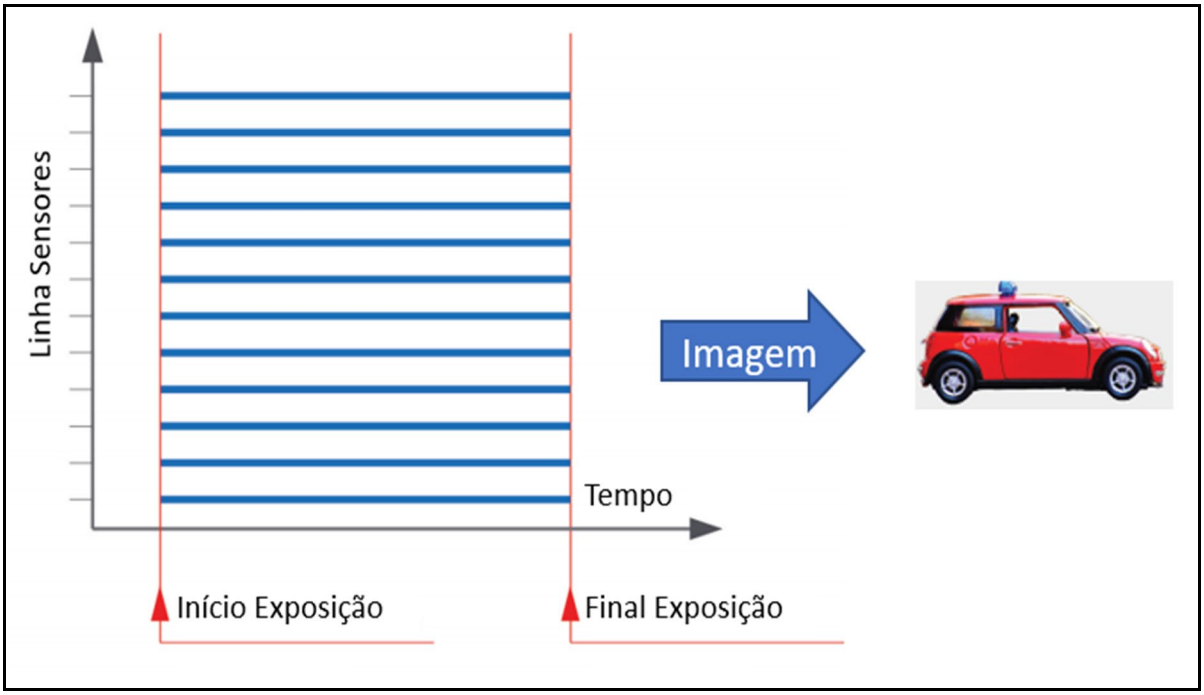


Figura 3 - Aquisição de Imagem usando câmera Global Shutter, Adaptado de: [16], [17].

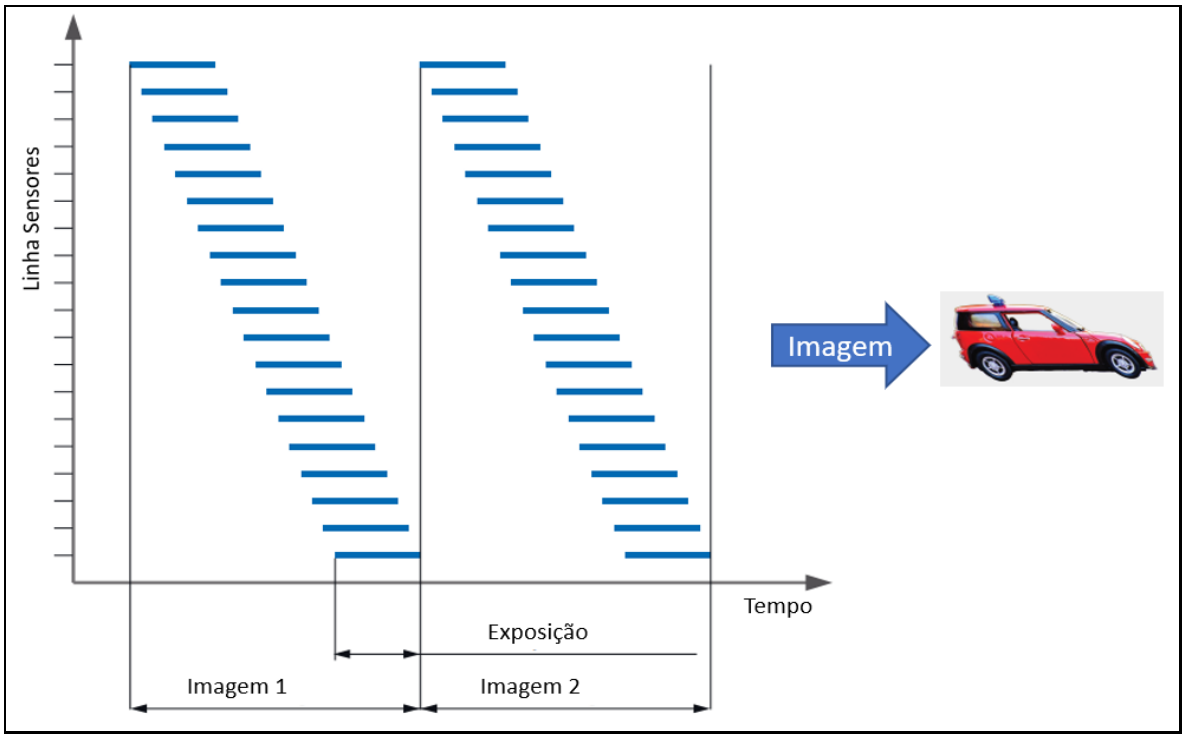


Figura 4 - Efeito do movimento usando uma câmera Rolling Shutter, Adaptado de [16], [17].

2.2 Representação de Imagens

2.2.1 Imagens no domínio espacial

Uma imagem pode ser representada por uma matriz retangular, sendo que o valor da imagem em um ponto *(x,y)* é expresso por *f(x,y)*, onde a coordenada *(x,y)* é dada por números inteiros. A função *f* representa a intensidade naquele ponto, ou seja, a intensidade é função das variáveis espaciais *x, y*. Assim, a seção do plano real que define a imagem é chamada de domínio espacial. Imagens representadas nesse sistema de coordenadas são chamadas de imagens no domínio espacial, possuindo um determinado número de linhas e colunas [19], [20].

A Figura 5 exemplifica uma imagem no domínio espacial, onde os valores de intensidade dos pixels estão no intervalo [0,1], trata-se de uma imagem em níveis de cinza, existindo apenas um valor de pixel por coordenada. A origem da imagem está localizada no canto superior esquerdo, a imagem possui uma resolução de *M* x *N* pixels.

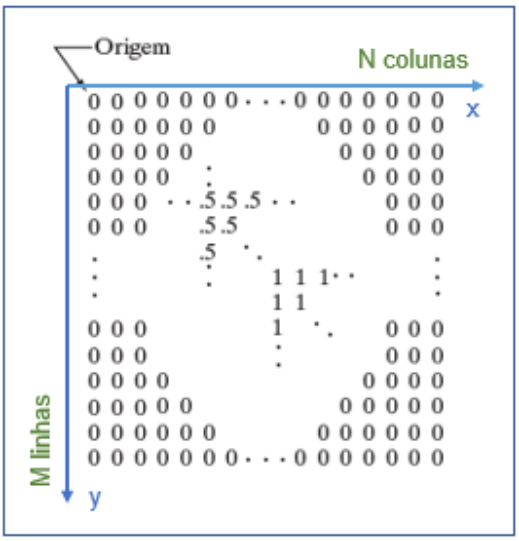


Figura 5 - Representação espacial de uma imagem em tons de cinza, Adaptado de [19].

2.2.2 Imagens no domínio da Frequência.

Uma imagem pode ser mapeada da sua representação no domínio espacial para o domínio da frequência através da Transformada Discreta de Fourier 2D (2D DFT – *2D Discrete Fourier Transform*). Operações no domínio da frequência podem

ser aplicadas de modo a melhorar o contraste, remover ruídos e suavizar as imagens [20].

A 2D DFT pode ser definida pela Equação 1 (Fonte [20]):

$$I(u,v) = \frac{1}{N.M}\sum_{x=0}^{N-1}\sum_{y=0}^{M-1} I(x,y).\exp[-i2\pi(\frac{xu}{N}+\frac{yv}{M})] \quad (1)$$

onde $I(x,y)$ é a intensidade do pixel nas coordenadas (x,y), a imagem possui N colunas e M linhas, $I(u,v)$ é o valor do pixel para as frequências espaciais u,v. O intervalo de u é definido por $[0,N-1]$, enquanto que o intervalo de v é definido por $[0,M-1]$. Finalmente, i refere-se à unidade imaginária $\sqrt{-1}$.

Os valores no domínio da frequência são valores complexos, podendo ser representados pela sua parte real (a) e parte imaginária (b) na forma da Equação 2 (Fonte [20]). A Imagem ainda pode ser descrita utilizando coordenadas polares na forma $(r,\ \alpha)$, onde r e α são definidos respectivamente pelas Equações 3 e 4 (Fonte [20]).

$$z = a + i.b \quad (2)$$

$$r = \sqrt{a^2+b^2} \quad (3)$$

$$\alpha = \mathrm{atan}(b,a) \quad (4)$$

Utilizando a Equação 3, pode-se representar a matriz de valores complexos na forma de um espectro de potência [20]. Segundo [21], a representação espectral de uma imagem é insensível a ruídos e invariante a translação, sendo ideal para detectar defeitos no ambiente de produção.

Após as operações no domínio da frequência a imagem pode ser novamente mapeada para o domínio espacial por meio da 2D DFT inversa, apresentada na Equação 5 (Fonte [20]).

$$I(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{M-1} I(u,v).\exp[i2\pi(\frac{xu}{N}+\frac{yv}{M})] \quad (5)$$

2.3 Redes Neurais Artificiais

2.3.1 Fundamentos

As redes neurais artificiais (NN – *Neural Networks*) são inspiradas no sistema nervoso humano. Assim como os neurônios são as unidades básicas de processamento no cérebro humano, o seu modelo simplificado, o neurônio artificial, corresponde à unidade básica das NN. Tais redes são capazes de realizar mapeamentos não lineares de alta complexidade, sendo atualmente utilizadas no desenvolvimento de poderosas ferramentas de processamento de informação, incluindo processamento de imagens e aprendizado de máquina [22].

O neurônio artificial pode ser visto como um elemento capaz de receber estímulos, que são as suas entradas, de diversos outros neurônios ou fontes e propagar sua única saída, dependendo dos estímulos recebidos e do seu estado atual. A Figura 6 e a Equação 6 (Adaptado de [22]) representam a descrição do neurônio artificial conforme apresentado.

$$y = f\left(\sum_{k=1}^{m} w_j x_j + b\right) \tag{6}$$

Considerando a definição do neurônio artificial apresentado na Figura 6, percebe-se a presença da função de ativação. Tal função tem como principais papéis: limitar a saída do neurônio e introduzir não linearidade ao modelo [22].

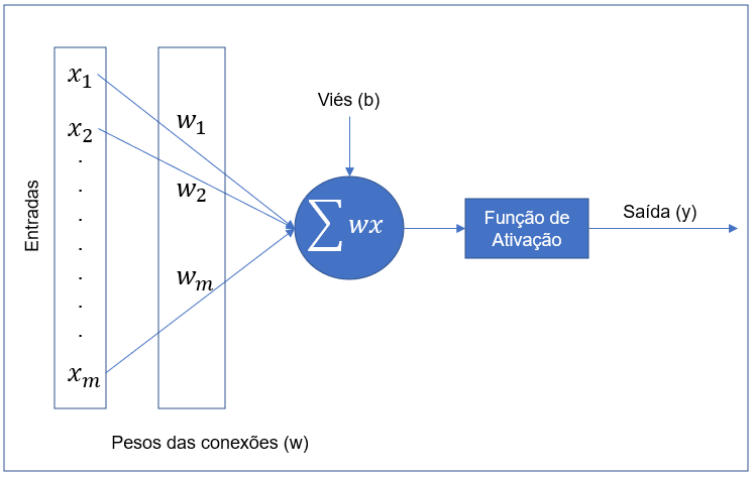


Figura 6 - Representação Neurônio Artificial. Adaptado de [22].

A interconexão e organização desses diversos neurônio formam uma NN, em que o conhecimento é adquirido por um processo chamado de aprendizado. O aprendizado corresponde à adequação dos pesos e limiares dos diversos neurônios da rede conforme os estímulos recebidos (entradas) [22]. A organização dos neurônios é feita em camadas: camada de entrada, camadas intermediárias e camada de saída. Uma rede com mais de uma camada intermediária é classificada como uma rede profunda (Figura 7), sendo que o seu treinamento é dito treinamento profundo (DL – *Deep Learning*)[14].

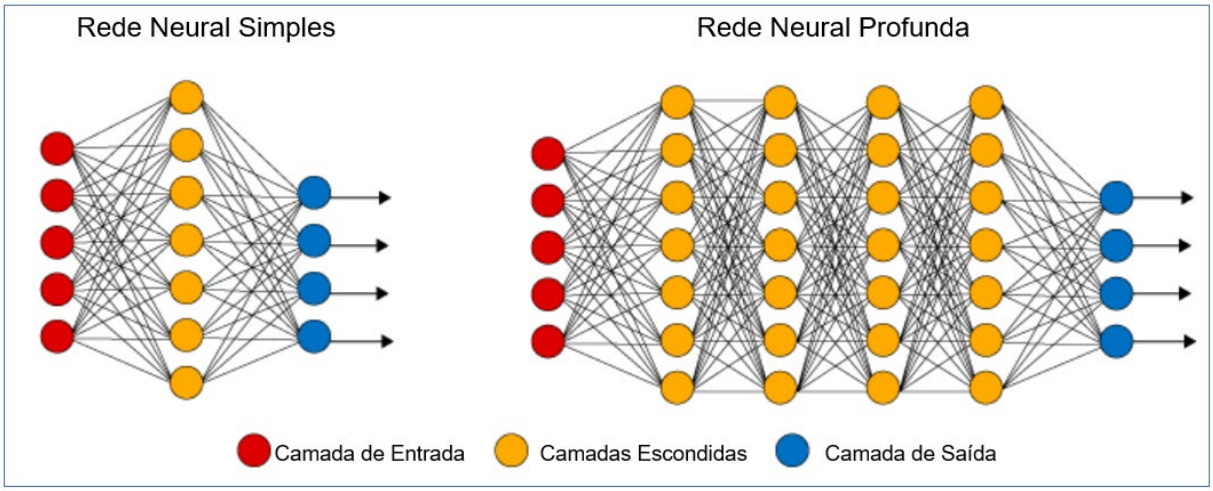


Figura 7 - Tipos de Redes Neurais. Adaptado de [23].

2.3.2 Redes Neurais Convolucionais (CNN – *Convolutional Neural Network*)

As CNN foram apresentadas pelo trabalho de Yann LeCun em 1998, a implementação e ideias envolvidas nesse trabalho foram inspiradas no trabalho de David H. Hubel e Torsten Weisel de 1968, no qual foram feitas observações e experimentos com o córtex visual de gatos [14], [15].

Desse modo, as CNNs são baseadas no conceito de campos receptivos, que são as conexões entre partes do sistema visual e determinados neurônios que processam essa informação. Assim, uma determinada região da imagem de tamanho *n x n* é ligada a um neurônio artificial, tal que essa janela da imagem recebe o nome de campo receptivo local. Ao deslocar essa janela sobre os pixels da imagem, as saídas dos neurônios formam o mapa de recursos (*features*), já os pesos e viés (*bias)* do campo receptivo local recebe o nome de filtro ou *kernel.* O

passo utilizado no deslocamento do filtro sobre a imagem é conhecido como *stride* [14], [24]. A Figura 8 representa de forma gráfica o que foi descrito no parágrafo.

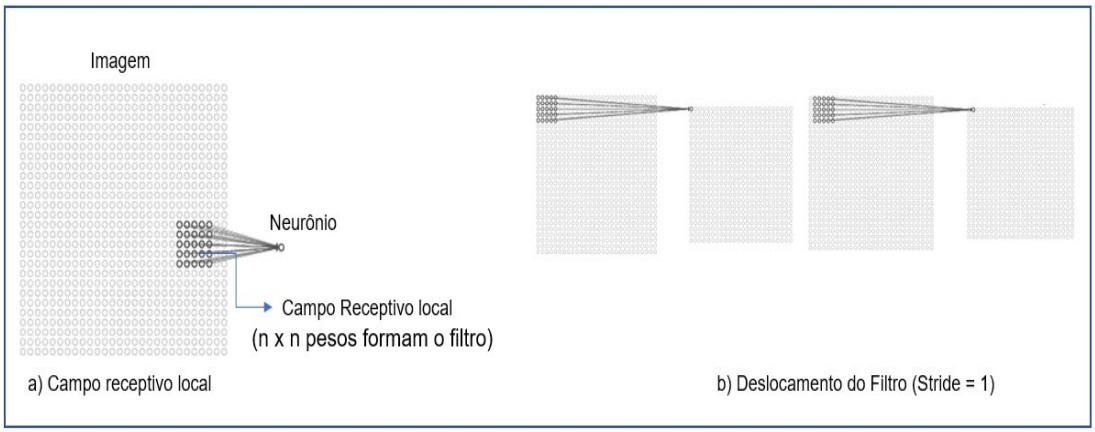


Figura 8 - Campo Receptivo e deslocamento do filtro. Adaptado de [24].

A aplicação do filtro, ou seja, matrizes com os pesos a serem aprendidos, atua de forma similar à aplicação de filtros clássicos do processamento de imagem, extraindo *features* da imagem ao qual são aplicados. O processo de aplicação de tais filtros é conhecido por convolução, o resultado da convolução de uma imagem com um filtro é um mapa de *features.*

Um dos problemas do uso da convolução é a perda de informação que pode ocorrer nas bordas da imagem. A técnica *padding* é usualmente utilizada para resolver esse problema, proporcionando também um maior controle no tamanho do mapa de *features* resultante. A técnica de *padding* pode ser vista como uma moldura de um determinado valor ao redor da imagem original, geralmente o valor considerado é 0, ou seja, tem-se a aplicação do *zero padding* [14], [25], [26].

Considerando o que foi descrito até o momento, a matriz resultante da convolução entre uma imagem e um determinado filtro terá sua dimensão estabelecida pela Equação 7 (Adaptado de [25], [27]):

$$M_o = 1 + \left[\frac{N + 2P - F}{S}\right] \tag{7}$$

onde M_0 é a dimensão da matriz resultante (mapa de *features* de tamanho M_0 x M_0), N é a dimensão da matriz de entrada (imagem N x N), F é o tamanho do filtro (F x F), P é o *padding* utilizado e S refere-se ao *stride*.

A Equação 8 (Fonte [25]), por sua vez, mostra como são calculados cada pixel do mapa de *features* resultante da convolução.

$$C(i,j) = (x * w)[i,j] = \sum_{m=0}^{(M_0-1)} \sum_{n=0}^{(M_0-1)} x[m,n].w[i-m,j-n] \quad (8)$$

onde C é o mapa de *features* gerado, x é a imagem, w é o filtro, $(*)$ representa a operação de convolução em 2 dimensões e M_0 é a dimensão do mapa de *features* calcula pela Equação 7. A Figura 9 representa graficamente a aplicação da Equação 8.

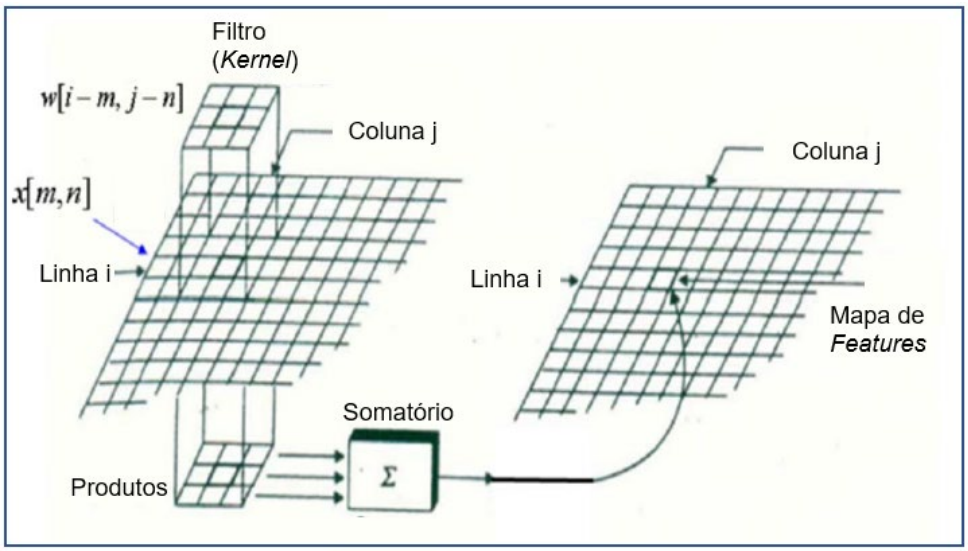


Figura 9 - Processo de Convolução, Adaptado de [25].

Logo após a camada de convolução é aplicada a função de ativação *ReLU* (*Rectified Linear Unit*) para ajustar os valores da saída. A função *ReLU* vem largamente sendo utilizada, pois apresenta um grande avanço em termos de velocidade e acurácia, quando comparada com outras função não-lineares, como a sigmoide e tangente hiperbólica [15]. A função ReLU é calculada de acordo com a Equação 9 (Fonte [28]), e a sua representação gráfica é apresentada na Figura 10.

$$f(x) = \max(0, x) \quad (9)$$

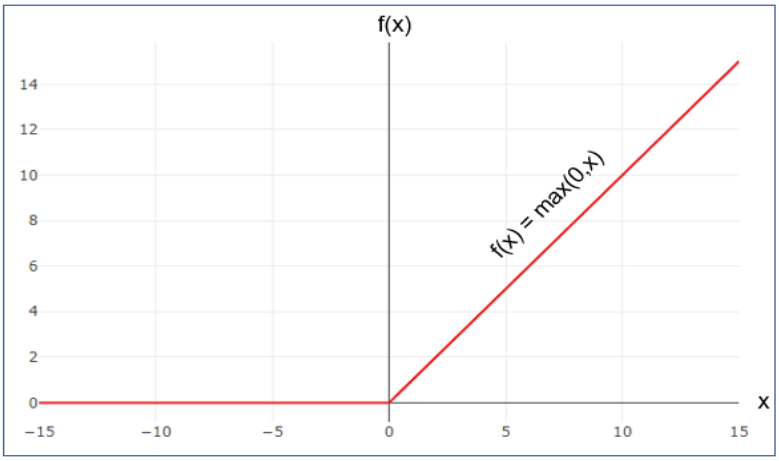


Figura 10 - Gráfico da função ReLU, Adaptado de [28].

Finalmente, para diminuir a complexidade do mapa para as próximas camadas é aplicada a técnica *pooling.* Com a aplicação do *pooling* diminui-se a resolução do mapa de *features*, sendo o *pooling* quase sempre aplicado no mapa de *features* ao invés da imagem original. Dentre os diversos métodos de *pooling* possíveis, o mais comum é o uso do *max-pooling.* Nesse caso, o mapa é dividido em regiões de um determinado tamanho e retorna-se o maior valor dentro de cada sub-região [14], [15].

Com os termos anteriormente definidos, pode-se dizer que as CNNs são formadas por diversas camadas convolucionais e camadas de *pooling* intermediárias. Ao final da arquitetura existe uma camada totalmente conectada de neurônios, camada esta semelhante às redes neurais tradicionais, onde cada *feature* do mapa tem uma conexão com cada neurônio dessa camada. Por fim, existe a camada de saída, onde o número de neurônios representa as possíveis classes as quais a imagem pertence; neste caso utiliza-se a função de ativação *softmax* para encontrar a probabilidade de a imagem pertencer a cada uma das classes. A Equação 10 (Fonte [14]) representa a função *softmax*.

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{N} e^{z_k}}, \quad j = 1, \ldots, N \tag{10}$$

onde z é o vetor com as saídas da última camada, N é o número de classes e j é a classe cuja a probabilidade está sendo calculada.

A Figura 11 resume os itens discutidos até o presente momento, exemplificando um uso clássico das CNNs que é a classificação de imagens.

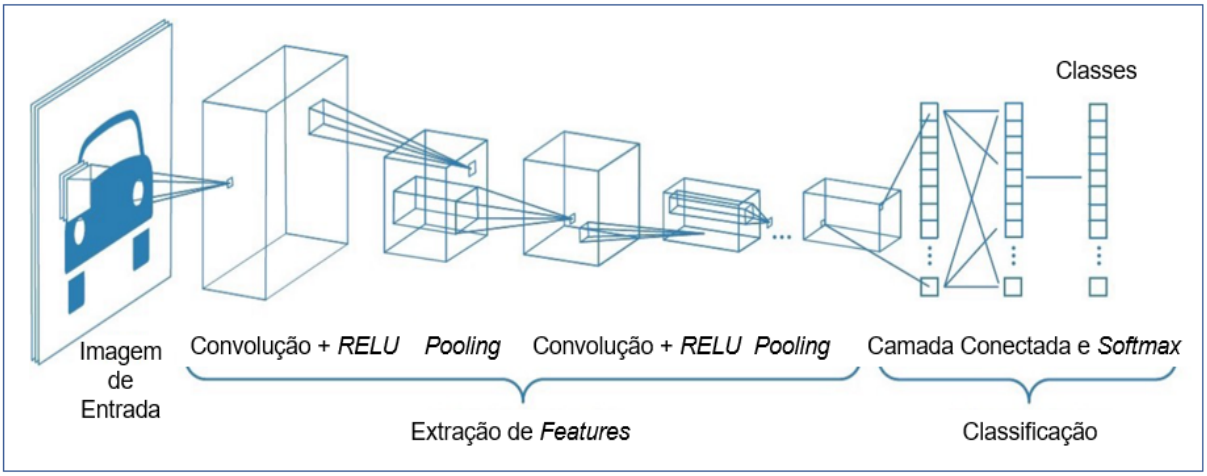


Figura 11 - Arquitetura geral de uma rede neural. Adaptado de [29].

2.3.3 Treinamento

O processo de treinamento é responsável por determinar os filtros, utilizados nas camadas convolucionais, e encontrar os pesos nas camadas totalmente conectadas. Esses valores de filtros e pesos, utilizados na arquitetura, tem por objetivo minimizar a diferença entre a predição de classes e os rótulos verdadeiros das imagens do conjunto de treinamento. O treinamento de uma CNN, representado graficamente pela Figura 12, corresponde a dois processos:

- *Forward propagation*: Nesta etapa a imagem passa por todos os elementos da arquitetura da rede e então calcula-se o desempenho do modelo, por meio da função de custo, sendo que esta recebe como entrada a saída da CNN e as imagens rotuladas.
- *Backpropagation*: processo onde os pesos e filtros são atualizados, de forma a minimizar a função de custo em relação ao conjunto de treinamento.

Esses processos são repetidos um determinado número de vezes, sendo esse número conhecido como época. A velocidade que os pesos e filtros são atualizados, na procura por uma melhor acurácia do modelo, é determinada pela taxa de aprendizado (lr – *learning rate*).

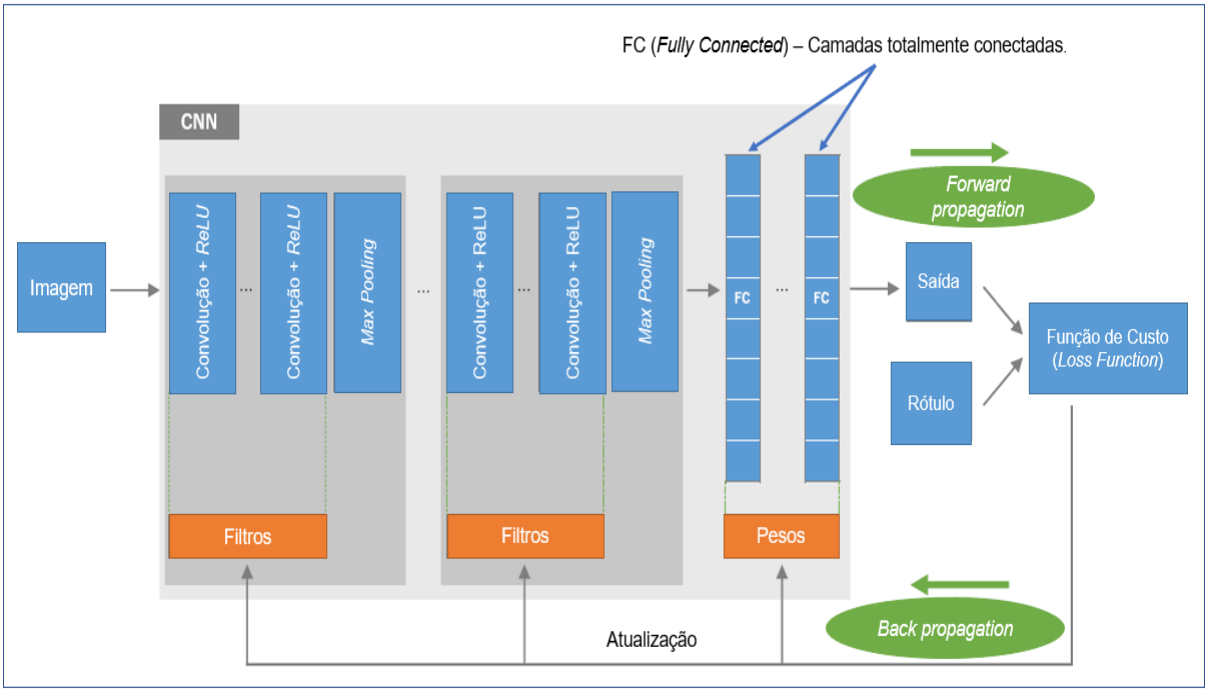


Figura 12 - Processo de Treinamento CNN (Adaptado de [28])

2.3.3.1 *Funções de Custo (Loss Function).*

O objetivo do treinamento é minimizar o erro da predição, no caso da CNN para classificação significa diminuir o erro da classificação em relação aos rótulos, previamente estabelecidos no conjunto de treinamento. Em outras palavras, a *loss function* mede a convergência entre as predições (saída da arquitetura CNN) e os rótulos verdadeiros [15], [28].

Existem diversas funções de custo que podem ser utilizadas no treinamento das CNN, sendo a sua escolha um dos hiper parâmetros que devem ser determinados de acordo com a aplicação. No presente trabalho, considerando as arquiteturas U-Net e VGG16 foram utilizadas as seguintes funções: *Binary Cross-Entropy* e *Categorical Cross-Entropy*.

Ambas as funções são derivadas da função *Cross-Entropy* (entropia cruzada), sendo esta utilizada para quantificar a diferença entre duas distribuições de probabilidade. Em uma CNN de classificação, o vetor de saída pode ser visto como uma distribuição de probabilidade. Assim, com o uso dessa função mede-se a distância entre a distribuição de probabilidade da predição e a distribuição verdadeira (dados rotulados do conjunto de treino).

No caso da classificação binária utiliza-se a função *Binary Cross-Entropy*. A Equação 11 (Adaptada de [30]) define essa função:

$$L_{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i \,.\log_2(p(x_i)) + (1 - y_i).\log_2(1 - p(x_i)) \quad (11)$$

onde N é a dimensão dos dados, y_i é o valor do rótulo (0 ou 1) para o i-ésimo dado e $p(x_i)$ é a predição realizada pelo modelo para o i-ésimo dado.

Já na classificação em categorias proposta pela rede VGG16, o uso da função de custo *Categorical Cross-Entropy* é mais comum. A Equação 12 (Adaptada de [31]) define essa função, devendo o conjunto de treino estar codificado no formato *one-hot*. Nesse formato, tem-se um vetor de *N* colunas, de modo que a coluna com valor 1 define a classe à qual a amostra pertence, e as demais colunas apresentam valor 0.

$$L_{CCE} = -\sum_{i=1}^{N} p(x_i)\log_2(q(x_i)) \quad (12)$$

onde $q(x_i)$ e $p(x_i)$ representam respectivamente a distribuição de probabilidade dos rótulos previstos pela CNN e a distribuição de probabilidade dos rótulos pré-determinados do conjunto de treinamento.

No caso do *autoencoder*, a função de custo utilizada foi o Erro Médio Quadrático (MSE – *Mean Squared Error*) que será definida com maiores detalhes na subseção 2.5.4.

No caso da GAN, a sua função de custo será detalhada na subseção 2.3.5 que apresenta os detalhes desse método.

2.3.3.2 Otimizadores (*Optimizers*)

O objetivo dos otimizadores no treinamento das CNN é atualizar os pesos e filtros de forma a minimizar a função de custo, melhorando assim a acurácia do modelo. Diversos algoritmos podem ser utilizados: *Gradient Descent, Stochastic*

Gradient Descent (SGD)*, Mini-Batch Gradient Descent, Adaptive Moment Estimation* (Adam), etc [15].

O algoritmo Adam foi o otimizador usado nesse trabalho, pois é um dos mais utilizados e efetivos, incorporando as vantagens propostas pelos algoritmos: *RMSProp* e *AdaGrad.* A principal diferença do Adam é que o mesmo não utiliza um único *lr*, cada peso da rede possui um *lr* próprio que é atualizado separadamente durante o processo de treinamento [32].

2.3.4 Aumento das dimensões espaciais

Em alguns métodos utilizados neste trabalho, é importante que a imagem reconstruída apresente as mesmas dimensões da imagem de entrada. O processo de convolução, utilizado pelas CNN, causam a diminuição da dimensionalidade. A Figura 13, mostra um exemplo de convolução onde a matriz resultante teria as dimensões 3x3, de acordo com a Equação 7.

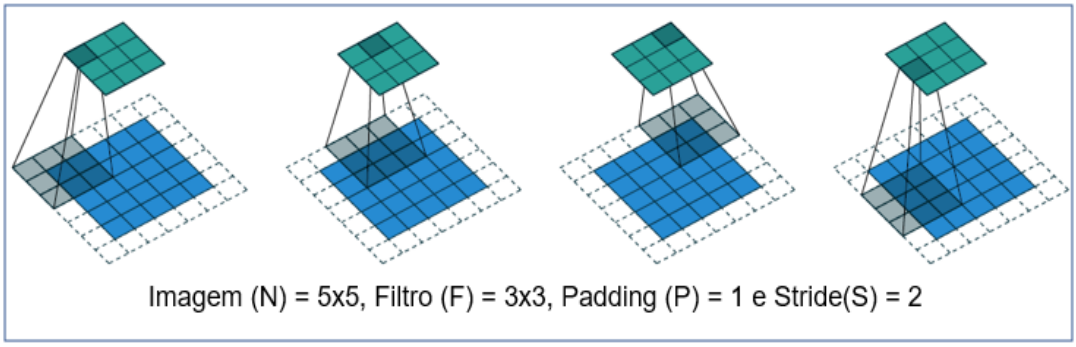


Figura 13 - Exemplo de uma operação de convolução (Adaptado de [27])

Nas subseções seguintes, serão descritos os métodos, considerados neste trabalho, para aumentar as dimensões espaciais.

2.3.4.1 ***UpSampling2D***

Dentre as diversas implementações possíveis para o *UpSampling2D*, neste trabalho utilizou-se a implementação padrão da biblioteca *Keras* que é baseada na interpolação do vizinho mais próximo (*Nearest Neighbor Interpolation*). Nesta técnica, não há a adição de novos valores, conforme mostrado na Figura 14.

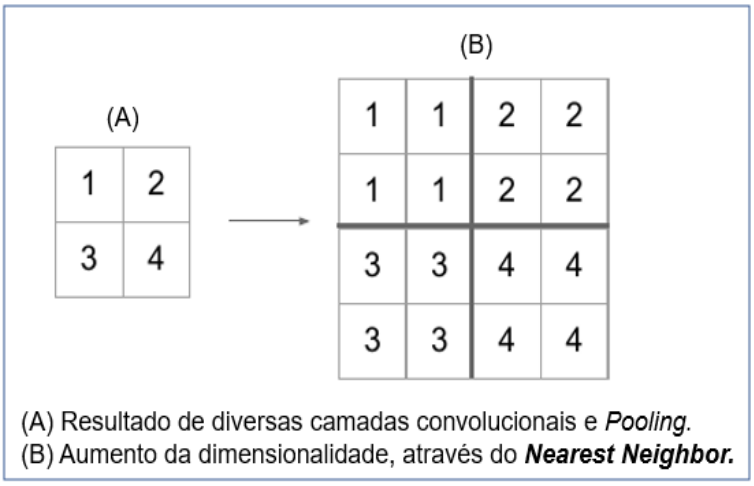


Figura 14 - Exemplo de Upsampling2D (Adaptado de [33]).

2.3.4.2 Convolução Transposta

A convolução transposta (*Transposed convolution*), também referenciada por deconvolução (*deconvolution*), também pode ser utilizada quando se deseja uma transformação na direção oposta da convolução [27]. Uma das vantagens do uso desse método, em relação ao *UpSampling2D,* é que os pesos do filtro podem ser aprendidos melhorando o processo de reconstrução.

A convolução transposta pode ser obtida a partir de uma operação de convolução, sendo necessárias modificações nos parâmetros de *padding* e *stride*. A Figura 15 exemplifica a convolução transposta a partir da convolução apresentada na Figura 13.

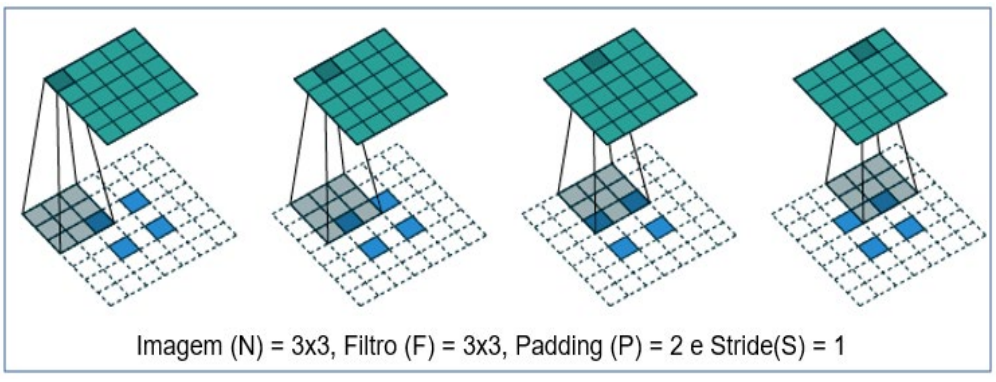


Figura 15 - Exemplo de uma operação de deconvolução (Adaptado de [27])

2.3.5 Autoencoders

Um *autoencoder* é uma técnica de aprendizado não supervisionado, sendo as vezes chamado também de aprendizado de representação. A principal função do *autoencoder* é reconstruir os dados de entrada após compactá-los em uma representação no espaço latente [14]. Essa rede neural é composta por dois processos, o do codificador (*encoder)* e do decodificador *(decoder)*. O codificador compacta a entrada de modo a extrair *features* que possuem alguma interpretação visual, enquanto que o decodificador reconstrói essa informação recriando os dados de entrada [15], [24].

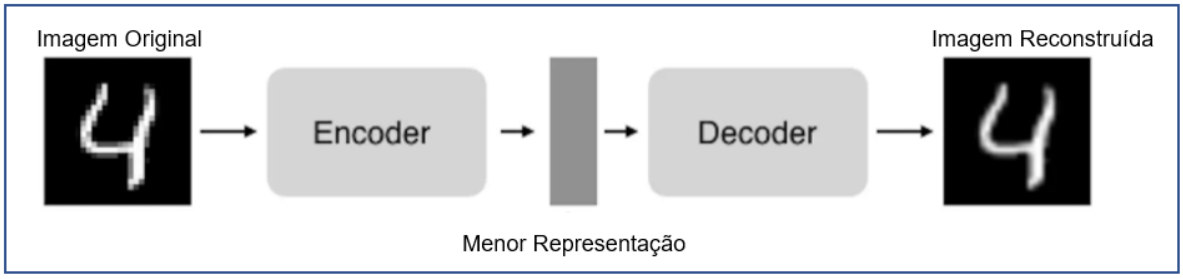


Figura 16 - Aplicação autoencoder em imagem do conjunto MNIST, Adaptado de [24].

Considerando a imagem da Figura 16, o *autoencoder* consegue aprender a menor representação da imagem (espaço latente), percebe-se assim possíveis usos do *autoencoder*, remoção de ruídos [24] e redução do efeito de serrilhamento (*anti-aliasing*). Além disso, o *autoencoder* é muito utilizado para a redução de dimensionalidade. No caso do presente trabalho, o *autoencoder* será utilizado na detecção de anomalias (*outliers*) [15], [34].

Considerando a detecção de anomalias, o *autoencoder* ao ser treinado, somente com as imagens sem anomalias, encontra um subespaço ótimo. Ao aplicar o *autoencoder* em uma imagem com anomalia a reconstrução da imagem terá um erro grande. Assim, ao aplicar-se um limiar no erro encontrado, é possível classificar as imagens como tendo anomalias ou não.

Dentre os diversos tipos de *autoencoder*, o *autoencoder* convolucional (*CAE – Convolutional AutoEncoder*) apresenta os melhores resultados na detecção de anomalias. Além disso, devido ao fato de possuir menos parâmetros, o seu

treinamento é feito em um menor tempo quando comparado com os *autoencoders* tradicionais [6], [34].

O CAE utiliza camadas convolucionais no *encoder* e camadas deconvolucionais na parte do *decoder*, conseguindo assim encontrar um espaço ótimo onde são capturadas as correlações não lineares entre as *features.* Em resumo, no CAE, as camadas convolucionais são responsáveis pela redução na dimensionalidade enquanto que as camadas deconvolucionais são aplicadas para reconstruir a imagem. O objetivo da fase de treinamento é minimizar o erro de reconstrução [6], [34]. A Figura 17 apresenta a topologia básica do CAE.

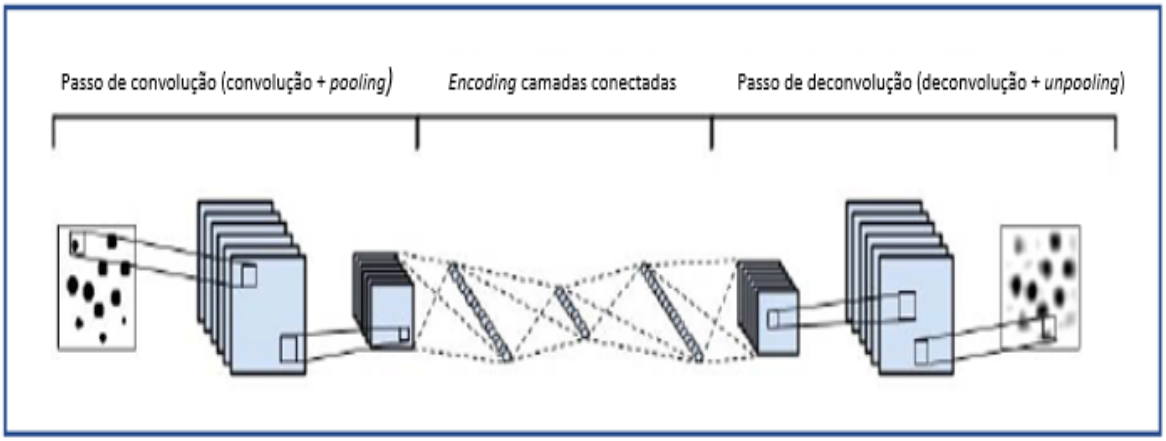


Figura 17 - Topologia CAE, Adaptado de [6].

2.3.6 Redes Adversárias Generativas (*GAN – Generative Adversarial Networks*)

Uma GAN é composta por dois modelos: um modelo gerador e um modelo discriminador. O modelo discriminador é formado por um algoritmo capaz de classificar um mapa de *features* de diversas dimensões. Por outro lado, o algoritmo generativo que representa o gerador é capaz de imitar distribuições de dados [24].

O discriminador tem por objetivo determinar se uma amostra foi gerada pela GAN ou é uma amostra real. O treinamento de uma rede GAN é uma competição entre esses dois elementos, onde o modelo generativo deve maximizar a probabilidade do discriminador cometer um erro [35], [36]. A Figura 18 representa o funcionamento de uma GAN.

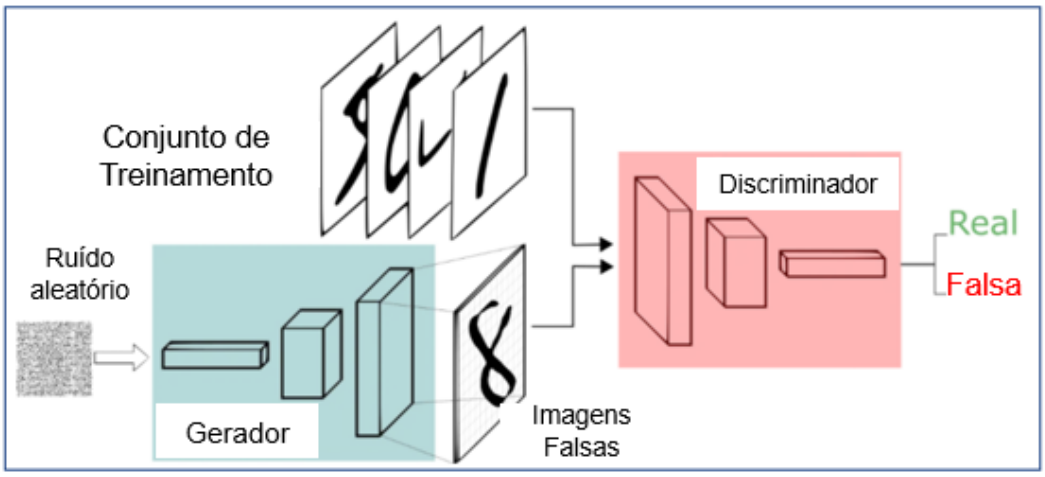


Figura 18 - Representação GAN, Adaptado de [24].

Os métodos de detecção de anomalias são baseados na distribuição dos dados que pode ser complexa e apresentar alta dimensionalidade. As redes adversárias generativas estão sendo usadas, com sucesso, para modelar esse tipo de problema. Desse modo, conforme os modelos baseados em GAN, propostos em [22] e [23], ao acrescentar um *encoder* à arquitetura é possível utilizar o novo modelo na detecção de anomalias de forma eficiente. Ainda conforme descrito em [38] e [37], as anomalias são detectadas a partir de um *score* composto pelo erro residual do discriminador e o erro de reconstrução do *encoder*. A Figura 19 representa o processo de detecção de anomalia descrito. Vale ressaltar que se trata de um treinamento não supervisionado, onde somente os dados sem anomalias são apresentados nessa fase.

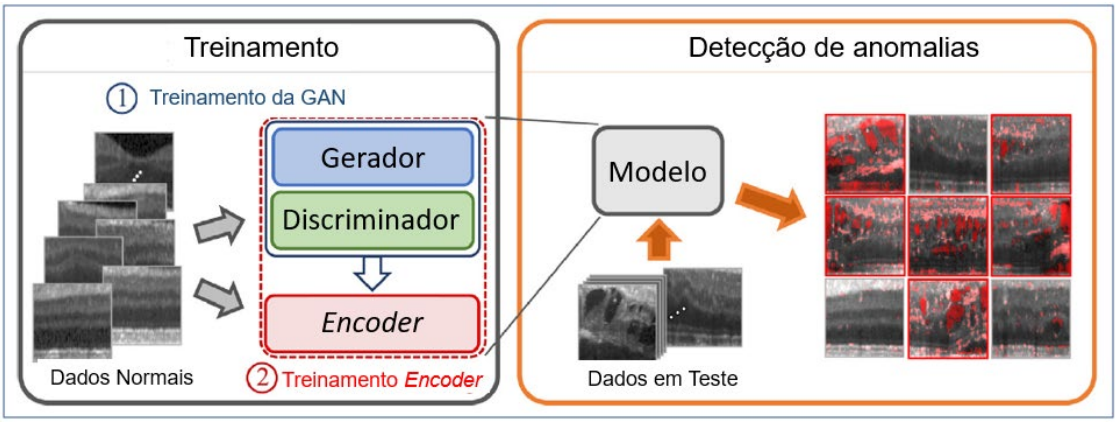


Figura 19 - Modelo para detecção de anomalias baseado em GAN, Fonte [38].

Outra forma de utilizar uma GAN na detecção de anomalias é considerar a saída do discriminador. No processo de treinamento, o discriminador aprende a reconhecer entre imagens geradas e imagens originais. Utilizando-se somente imagens sem anomalias no treinamento, o discriminador irá aprender com tais imagens e, portanto, irá conseguir detectar as imagens com anomalias [11].

Retornando aos fundamentos da GAN, o jogo *minimax*, cujos participantes são o discriminador e o gerador, tem a Equação (13) (Adaptado de [39]) como função objetivo.

$$\min_{G}\max_{D} V(D,G) = \mathop{\mathbb{E}}_{x\sim\mathbb{P}_r}[\log(D(x))] + \mathop{\mathbb{E}}_{\tilde{x}\sim\mathbb{P}_g}[\log(1-D(\tilde{x}))] \tag{13}$$

onde $\mathbb{P}_r$ é a distribuição sobre as amostras reais x, $\mathbb{P}_g$ é a distribuição do gerador definida por $\tilde{x} = G(z)$, z representa o ruído (distribuição uniforme) que alimenta o gerador, $D(x)$ é a saída do discriminador para uma amostra real e $D(\tilde{x})$ é a saída do discriminador para um dado gerado. As notações iniciais da Equação 13, definem que o gerador deve minimizar a função, enquanto que o discriminador deve maximizá-la.

Segundo [35] a Equação 13 pode não fornecer gradientes suficientes para o gerador ter um bom aprendizado, propondo que o treinamento do gerador deva maximizar $\log(D(\tilde{x}))$. Além disso, outros problemas são conhecidos no treinamento da GAN e catalogados na literatura [40]:

a) Oscilação na função de custo: embora alguma oscilação seja comum, em alguns casos a função de custo tanto do gerador quanto do discriminador não estabiliza com o passar das épocas. A Figura 20 (a) exemplifica um modelo que o treinamento foi bem sucedido, enquanto Figura 20 (b) mostra uma GAN com oscilações.
b) Colapso do Modelo: ocorre quando o gerador encontra um número de imagens que consegue enganar o discriminador, ficando preso nesse pequeno conjunto de imagens e não aprendendo uma representação mais robusta.
c) Gráfico da função de custo pouco informativa: não existe uma correlação entre a qualidade das imagens geradas e a função de custo do gerador, nos modelos de DL geralmente o objetivo é minimizar a função de custo.

Porém na Figura 20 (a) a função de custo do gerador aumenta com o número de épocas.

d) Hiper parâmetros: ambos o gerador e discriminador possuem uma série de parâmetros para serem ajustados, sendo a GAN bem sensível a qualquer mudança. Então, um processo de tentativa e erro é necessário nessa busca pelos melhores valores, exigindo um bom entendimento do mecanismo da GAN e de tempo para o treinamento e avaliação dos modelos.

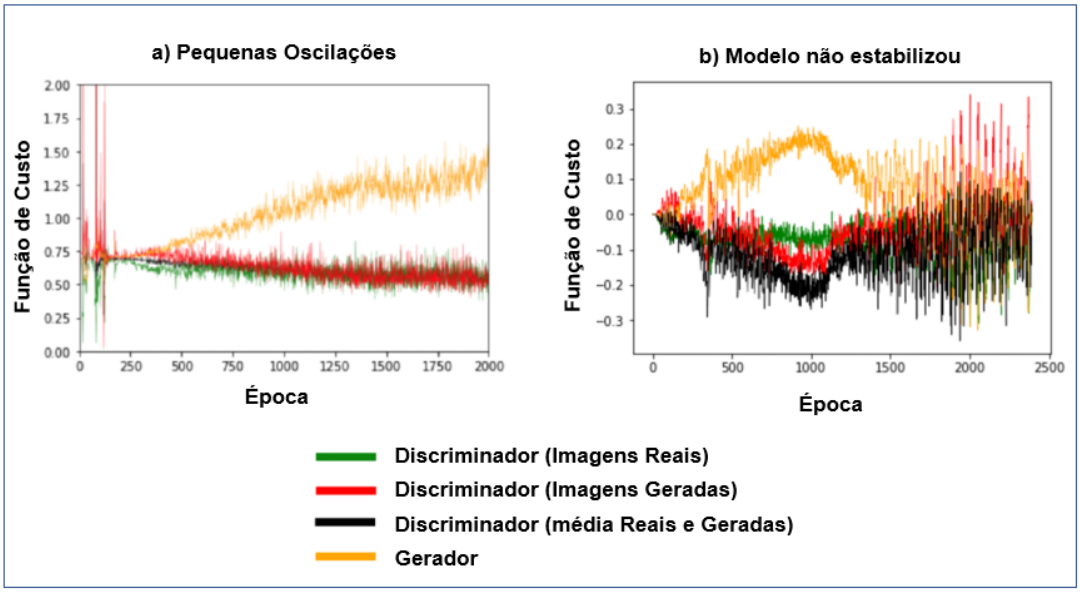


Figura 20 - Modelo estável x Modelo com Oscilações (Adaptado de [40])

Visando contornar os problemas apresentados, várias modificações estão sendo propostas, como as apresentadas nas próximas subseções.

2.3.6.1 WGAN (Wasserstein GAN)

Com algumas mudanças no modelo conseguiu-se estabelecer uma função de custo que faz a correlação entre a convergência do gerador e a qualidade da imagem, além de melhorar a estabilidade no processo de otimização [40].

Na WGAN os rótulos das imagens reais devem ser 1, enquanto que a imagens geradas devem ser rotuladas com -1 (na GAN as imagens geradas eram rotuladas com 0). A função sigmoide do discriminador deve ser removida, assim as predições serão representadas por um número no intervalo [-∞, +∞].

A saída do crítico pode resultar em valores muito grandes, fato que deve ser evitado nas redes neurais. Desse modo, foi adicionado na rotina de treinamento um parâmetro restritivo (*clipping)*, forçando os pesos do crítico estarem no intervalo de [-0,01; 0,01], desse modo o crítico irá obedecer a restrição de *Lipschitz* [41].

Com as mudanças propostas, a função objetivo do jogo *minimax* passa a ser representada pela Equação 14 (Fonte [39]):

$$\min_{G} \max_{D \in \mathcal{D}} V(D,G) = \mathop{\mathbb{E}}_{x \sim \mathbb{P}_r}[(D(x))] + \mathop{\mathbb{E}}_{\tilde{x} \sim \mathbb{P}_g}[(D(\tilde{x}))] \tag{14}$$

onde $\mathcal{D}$ é o conjunto de funções 1-*Lipschitz* e os demais itens da equação são os mesmos apresentados na Equação 13.

2.3.6.2 WGAN-GP (*Wassertein* GAN-*Gradient Penalty*)

No próprio artigo original da WGAN [40], [41], os autores admitem que o uso do parâmetro restritivo, que faz o crítico atender a restrição de *Lipschitz,* pode prejudicar o processo de aprendizado do gerador ao produzir melhores resultados.

Assim, a WGAN-GP foi proposta para melhorar a capacidade da WGAN em aprender *features* complexas. Para converter a WGAN em WGAN-GP, 3 modificações foram propostas:

- Inclusão de um termo chamado penalidade de gradiente (*gradiente penalty*) no crítico;
- Remover o uso do fator restritivo (*clipping*).
- Remover o uso de camadas de *batch normalization* [24] no crítico.

Com a inclusão da penalidade de gradiente, a função de custo utilizada na WGAN-GP é apresentada na Equação 15 (Fonte [39]).

$$L = \mathop{\mathbb{E}}_{\tilde{x} \sim \mathbb{P}_g}[D(\tilde{x})] - \mathop{\mathbb{E}}_{x \sim \mathbb{P}_r}[D(x)] + \lambda \mathop{\mathbb{E}}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2] \tag{15}$$

onde λ é o coeficiente de penalidade e $\mathbb{P}_{\hat{x}}$ é a amostragem feita ao longo de linhas retas entre pares de pontos amostrados das distribuições $\mathbb{P}_g$ e $\mathbb{P}_r$.

2.3.7 *Backbones*

Muitos dos modelos de redes neurais que serão utilizados no trabalho podem ser baseados em *backbones*, por exemplo: o *encoder* utilizado no *autoencoder* pode ser baseado em uma rede *InceptionV3*. A U-Net, a ser utilizada na segmentação da PCB, também pode ser baseada em *backbones* como por exemplo: *VGG, ResNet, Inception, MobileNet, EfficientNet*, etc [42].

Quando o termo *backbone* é utilizado no contexto das CNN, geralmente o mesmo se refere às camadas de uma determinada arquitetura utilizada para extrair *features* básicas. Tais camadas, são originalmente definidas para a o uso na classificação de imagens, sendo pré-treinadas em conjuntos de imagens conhecidos e utilizados como *benchmark*, tal como o conjunto ImageNet [43]. Dentre muitas arquiteturas de CNN, utilizadas como *backbone*, pode-se citar por ordem de popularidade: *AlexNet, VGG-19, GoogLeNet, ResNets, Inceptions e DarkNet.* [44].

2.3.8 *Transfer Learning*

As camadas das CNN são organizadas de forma a existir uma espécie de divisão de tarefas. As *features* extraídas pelas primeiras camadas representam características primitivas das imagens como por exemplo linhas, bordas ou cantos. Por outro lado, as camadas finais são responsáveis por extrair *features* mais complexas, que em geral estão relacionadas com o significado semântico da classificação, proposta pelo treinamento [15].

Nesse cenário, faz sentido o uso de modelos pré-treinados em novos problemas de classificação. Um determinado número de camadas e seus respectivos pesos são utilizados em uma nova CNN sendo que a camada de saída é reprojetada e retreinada de acordo com o problema a ser resolvido. Esse processo de reutilização de modelos pré-treinados recebe o nome de TL (*transfer learning*). Segundo [15] essa estratégia é tão popular que é muito raro que o treinamento de novos modelos não faça uso do TL.

Em [45] são listadas 2 motivos pelos quais o uso do TL é tão popular:

- O treino de uma CNN requer grandes quantidades de dados rotulados, exigindo especialistas em algumas áreas, por exemplo, medicina e eletrônica;

- Durante o treinamento podem ocorrer problemas de convergência, exigindo o ajuste de parâmetros da arquitetura e configuração do processo de aprendizado.

Conforme dito anteriormente, no TL os pesos das camadas CNN são mantidos, sendo necessário treinar ao menos a última camada. Porém, em alguns casos existe uma grande diferença entre a aplicação usada no treinamento inicial e a aplicação atual, desse modo pode-se usar o processo de *fine-tuning.* Esse processo consiste em escolher *n* camadas que serão retreinadas, usando os dados relativos à aplicação atual, de modo a alcançar o desempenho desejado [45]. Os pesos iniciais são mantidos, ao invés de se utilizar pesos inicializados de forma aleatória.

2.4 Segmentação e Registro da PCB

2.4.1 Abordagem Tradicional

A extração de *features* é um dos passos fundamentais no reconhecimento e identificação de objetos. Uma *feature* pode ser entendida como um padrão específico capaz de ser monitorado e comparado [46].

Uma outra definição de *feature* é a junção de pontos de interesse (*keypoints*) e descritores. Os *keypoints* são definidos como as intensidades da imagem ao redor do ponto e um descritor é um vetor com as principais propriedades do *keypoint* [20].

O algoritmo SIFT (*Scale Invariant Feature Transform*) é um dos melhores algoritmos para a extração de *features*, possuindo também uma boa capacidade de selecionar as principais *features* para melhor identificar os objetos de interesse [4]. O SIFT é composto por 4 passos [47]:

a) Detecção de extremos em diferentes escalas de intensidade de pixel: a imagem é borrada usando um filtro Gaussiano de diferentes tamanhos (σ) gerando assim imagens em diferentes escalas. Então, calcula-se a diferença entre as diversas imagens por meio do processo conhecido como DoG (*Difference of Gaussian*), procurando por extremos nas imagens de diferentes escalas que são potenciais *keypoints*;
b) Localização dos *keypoints*: os potenciais *keypoints* são refinados utilizando expansões em séries de Taylor e matrizes Hessianas, retornando assim os pontos de interesse mais relevantes;

c) Definição de Orientação: para cada um dos pontos refinados no processo anterior é atribuída uma orientação, fazendo que o algoritmo seja invariante a rotação da imagem;
d) Descritores do *keypoint*: através dos 16x16 vizinhos do *keypoint* calcula-se um vetor de 128 valores que representa o descritor de cada um dos *keypoints* encontrados.

A partir das *features* extraídas pelo SIFT, é necessário encontrar a correspondência entre as *features* da imagem de referência e as *features* da imagem a ser registrada e através destas promover transformações na imagem em análise. Neste contexto escolheu-se a homografia, que pode ser representada por uma matriz 3x3 de transformação, sendo que a relação entre um ponto (x_1, y_1) da imagem 1 com o ponto (x_2, y_2) da imagem 2 é dada pela Equação 16:

$$\begin{bmatrix} x1 \\ y1 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00}\ h_{01}\ h_{02} \\ h_{10}\ h_{11}\ h_{12} \\ h_{20}\ h_{21}\ h_{22} \end{bmatrix} \begin{bmatrix} x2 \\ y2 \\ 1 \end{bmatrix} \tag{16}$$

A matriz de homografia (H) é calculada pelo método de RANSAC (*Random Sample Consensus*) [20], que é uma técnica iterativa para estimar parâmetros de um modelo matemático, ou seja, aqui a matriz de homografia é o modelo e os parâmetros são calculados por meio dos *keypoints* das imagens retornados pelo SIFT [20].

2.4.2 Segmentação Semântica

A segmentação semântica é um processo em que é atribuída uma classe para cada pixel da imagem, sendo que as classes se referem aos componentes a serem segmentados e o fundo da imagem (*background*). No caso do objeto deste trabalho, para a segmentação dos *brackets*, existirão duas classes: *bracket* e esteira [7].

O método DL escolhido para executar a segmentação semântica é a U-Net. Essa rede foi inicialmente projetada para trabalhar com imagens médicas. Porém na descrição apresentada em [48], o problema em estudo nesse trabalho assemelha-se com a tratativa utilizada nas imagens médicas:

- Pequena base de dados para treinamento;

- Necessidade do uso do aumento da base de dados (DA - *Data Augmentation*);
- Variações na translação e rotação da PCB em relação à esteira, proporcionando a mesma regra para o DA conforme utilizada em [48].

A U-Net recebe esse nome porque é formada por duas partes chamadas de caminho de contração (*encoder path*) e caminho de expansão (*decoder path*). O caminho de contração é formado por diversas aplicações de convoluções 3x3 com função de ativação *ReLU* e operações de *max pooling* 2x2, diminuindo assim a resolução espacial da imagem, e é nessa fase que são capturadas as informações de contexto. Cada etapa do caminho de expansão é formada por um *upsampling* seguido por uma convolução 2x2, o resultado dessa operação é concatenado com as *features* correspondentes encontradas no caminho de contração. Por fim tem-se uma convolução 3x3 com ativação *ReLU*. Essa expansão simétrica cria a localização precisa das classes, a arquitetura descrita é apresentada na Figura 21.

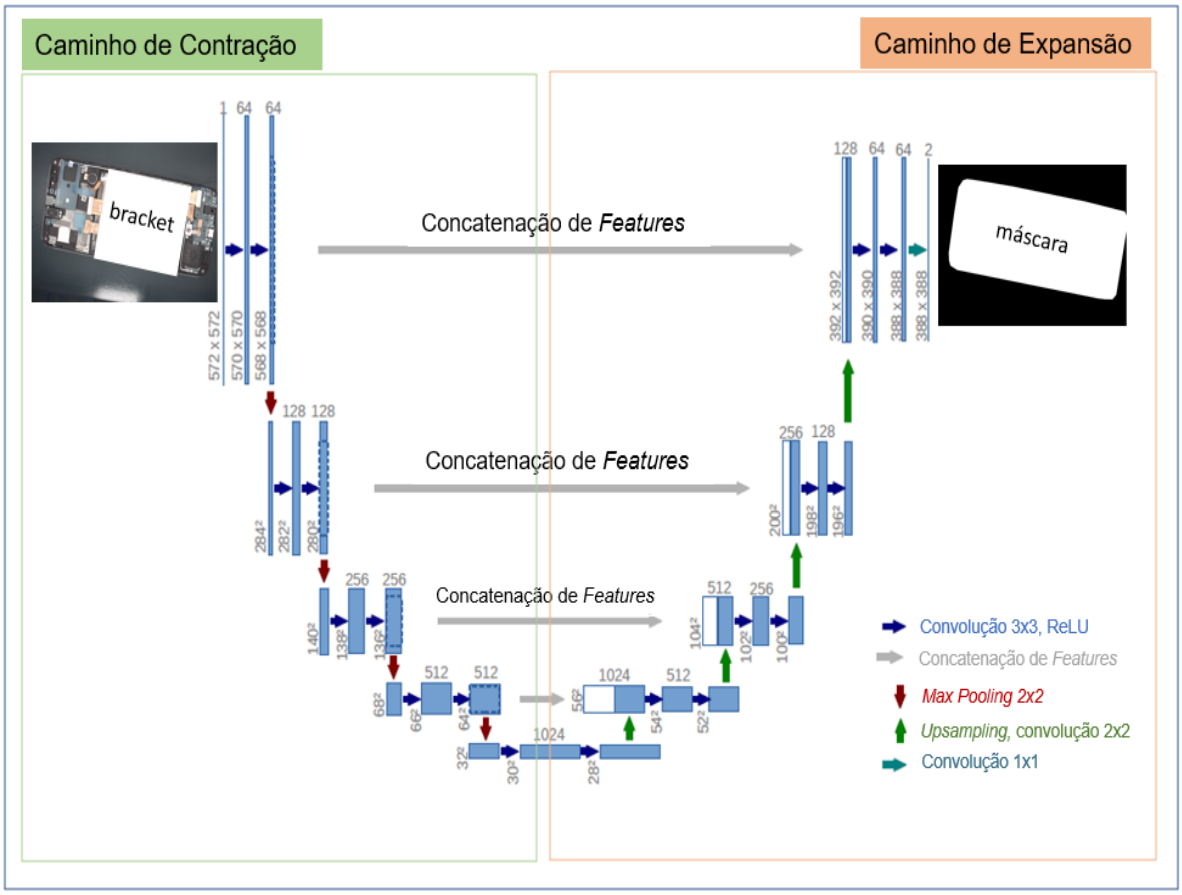


Figura 21 - Arquitetura U-Net, Fonte Adaptado de [48]

2.4.3 *Watershed*

A *watershed* morfológica apresenta muitos dos conceitos definidos em três métodos tradicionais: detecção de bordas, limiarização e crescimento de região. Nessa abordagem, são encontradas as fronteiras de segmentação conectadas, resultando em segmentações mais estáveis quando comparadas aos demais métodos tradicionais [19].

No conceito de *watershed*, as imagens em tons de cinza (*grayscale*) são interpretadas como relevos em uma representação topográfica. Desse modo, os valores de cada *pixel* da imagem, por exemplo no intervalo de 0 a 255 (considerando 8 bits), são vistos como as alturas dos elementos do relevo [49]. Considerando a elevação, os pontos do relevo podem ser classificados em 3 tipos: pontos que representam mínimo regional; pontos em que se uma gota de água fosse despejada sobre ele, esta desceria até um ponto de mínimo; e pontos onde a gota despejada teria a mesma probabilidade de cair em mais de um ponto de mínimo. Esses pontos são denominados linhas de divisão ou linhas de *watershed* [19].

Desse modo, a segmentação *watershed* pode ser visualizada baseando-se na seguinte analogia. Em cada um dos pontos mínimos será injetada água em uma taxa constante, quando a água acumulada em cada *watershed* estiver prestes a se juntar, constrói-se uma barreira impedindo a fusão. Interrompe-se o fluxo de água, quando somente os limites das barragens estiverem acima da linha d'água. Esses limites representam as linhas de segmentação no algoritmo de *watershed* [19].

Os algoritmos, baseados na analogia apresentada, costumam causar super segmentação (*oversegmentation*) principalmente devido a presença de ruídos nas imagens [49]. Visando evitar esse problema, o presente trabalho irá fazer o uso da função *watershed* da biblioteca *Opencv* [50]. Esta função é implementada considerando marcadores, que definem as regiões onde é possível determinar que não exista a construção de barreiras.

A Figura 22 apresenta um exemplo da segmentação usando *watershed* e marcadores. Neste exemplo, a *watershed* foi utilizada para melhorar a segmentação alcançada com o uso da U-Net. A partir do resultado da segmentação semântica foi possível estabelecer os marcadores: componente (região onde tem-se certeza que pertence ao componente), *background* (região onde sabe-se que não faz parte do componente) e região desconhecida (onde o algoritmo irá atuar).

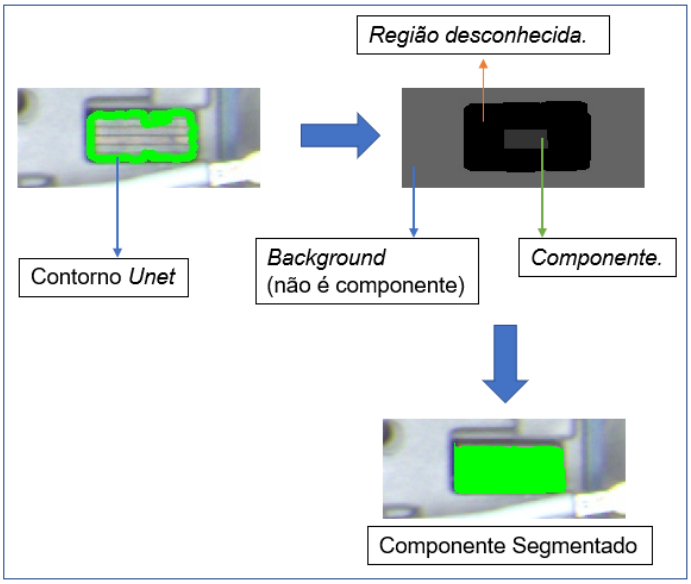


Figura 22 - Watershed com o uso de marcadores.

2.5 Métricas

Nas subseções seguintes são apresentadas as métricas utilizadas neste trabalho. As demais citadas nos artigos da revisão não serão discutidas.

2.5.1 Acurácia

A acurácia é uma das métricas mais utilizadas. A Equação 17 mostra a sua definição.

$$Acurácia = \frac{Número\ corretos\ de\ predições}{Total\ de\ predições\ feitas} \quad (17)$$

A acurácia pode trazer a falsa sensação que o modelo é suficiente para resolver um determinado problema. Por exemplo, na avaliação de anomalias proposta neste trabalho, existem alguns processos para garantir a qualidade dos *brackets*, como por exemplo o ICT (*In Circuit Test*). Assim, mesmo que o percentual de placas defeituosas na produção seja baixo, um algoritmo que não consegue identificar os defeitos pode ter uma acurácia alta.

2.5.2 Matriz de Confusão

É uma matriz que descreve o desempenho do sistema. Analisando a matriz é possível determinar: Verdadeiros Positivos (*TP – True Positives*), Verdadeiros Negativos (*TN – True Negatives*), Falsos Positivos (*FP – False Positives*) e Falsos Negativos (*FN – False Negatives*). A Tabela 1 apresenta um exemplo de Matriz que pode ser utilizado na avaliação dos resultados do presente trabalho.

Tabela 1 - Exemplo de Matriz de Confusão

	N = 100	**Predição**	
		Sem Defeito	Defeituoso
Classificação Verdadeira	Sem Defeito	90 (TN)	2 (FN)
	Defeituoso	5 (FP)	3 (TP)

2.5.3 IoU (Intersection Over Union) Score

É uma técnica de avaliação onde a saída do detector de objetos é a predição de um *bounding box*, ou seja, são as coordenadas do objetivo que está sendo segmentado pelo modelo proposto.

Para o cálculo desse índice é preciso conhecer as coordenadas reais do objeto a ser detectado, que podem ser encontradas manualmente com a ajuda de alguma ferramenta de anotação, e as coordenadas retornadas pela predição executada pelo modelo. A partir das coordenadas criam-se 2 máscaras, a primeira representando a localização real do objeto (máscara A), e a segunda a localização predita pelo modelo (máscara B).

A máscara é uma imagem onde os pixels referentes a localização do objeto possuem valor 1, enquanto que os demais pixels possuem valor 0. A partir das máscaras e operações *bitwise*, o índice IoU é calculado (Equação 18):

$$IoU = \frac{E\ (máscara\ A, máscara\ B)}{OU\ (máscara\ A, máscara\ B)} \tag{18}$$

O uso do *IoU Score* não é uma abordagem nova, sendo utilizado na avaliação de diversos algoritmos, tais como U-Net [48] e YOLO (*You Only Look Once*) [51].

2.5.4 MSE (*Mean Squared Error*)

O Erro Quadrático Médio (MSE – *Mean Squared Error*) é definido pela Equação 19 (Fonte: [52]). O MSE será utilizado neste trabalho, principalmente como função de custo no treinamento do *autoencoder*, além de avaliar a presença de anomalias utilizando essa mesma técnica de rede neural.

$$MSE = \frac{1}{N}\sum_{i=0}^{N-1}(y_i - p_i)^2 \tag{19}$$

onde N é o número total de amostras; y_i é o valor esperado da saída da função sendo avaliada e p_i é o valor retornado pelo modelo (predição).

2.5.5 Escore-z (*z-Score*)

O Escore-z pode ser interpretado como o número de desvios padrão, que o valor normalizado de uma variável aleatória, se afasta da média da sua distribuição de frequências. O Escore-z é calcula pela equação 20 (Fonte [22]).

$$\text{Escore-z} = \frac{x - \overline{x}}{\sigma} \tag{20}$$

onde: x é o valor da amostra, $\overline{x}$ é a média e σ representa o desvio padrão.

Neste trabalho os gráficos de Escore-z serão utilizados para determinar a presença de valores discrepantes (*outlier*) e verificar quantos desvios padrões podem ser considerados de modo a encontrar as anomalias.

2.5.6 K-médias (*K-means)*

O algoritmo K-means é um algoritmo de agrupamento. Neste tipo de algoritmo, os objetos (representados por vetores de *features*) são organizados em grupos. Os

grupos são formados de forma que exista homogeneidade (coesão interna) com isolamento externo, ou seja, os elementos dentro do grupo são similares entre si e a distância entre elementos de grupos diferentes é significativa.

O treinamento nos métodos de agrupamento é dito não supervisionado, pois dados do treinamento não são rotulados, como é feito, por exemplo, nas CNNs de classificação.

Um dos algoritmos mais usados para o agrupamento é o *K-means*. Esse algoritmo recebe como entrada um parâmetro *K,* que representa o número de grupos a serem formados. O algoritmo tende a formar grupos com alta similaridade intragrupo, sendo avaliados por meio do valor médio dos objetos do grupo, que recebe o nome de centro de gravidade ou centroide. Assim, ao final do processo de agrupamento, cada objetivo irá pertencer ao grupo do centroide mais próximo a ele. A medida de distância utilizada nesse trabalho é a distância Euclidiana.

Para avaliar a qualidade do agrupamento, neste trabalho, será utilizado o coeficiente de silhueta. Este índice informa quão bem agrupados os objetos estão, com os valores possíveis dentro intervalo de [-1;1]. Se o índice possui valor próximo de 1, indica que houve um bom isolamento entre os grupos (*clusters*), por outro lado, valores negativos indicam que não existe um isolamento entre os grupos formados havendo sobreposição.

A Equação 21 (Fonte [22]) é utilizada para o cálculo do coeficiente de silhueta.

$$S = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{|g_i|}\sum_{j=1}^{|g_i|}\frac{b(j)-a(j)}{\max\{a(i),b(j)\}} \tag{21}$$

onde k é o número de grupos; $|g_i|$ é o número de objetos no i-ésimo grupo; $a(i)$ é a distância média do j-ésimo objeto do grupo g_i aos objetos do mesmo grupo; e $b(j)$ é a menor distância média do j-ésimo objeto do grupo g_i aos objetos dos outros grupos.

3 Revisão Bibliográfica

Nos parágrafos da introdução foi ressaltada a importância da inspeção visual nas PCB de produtos eletrônicos. Desse modo, o uso da AOI *(Automated Optical Inspection*) vem sendo estudado e existem diversos trabalhos sobre o assunto. Tais trabalhos propõem diversos métodos de inspeção visual de modo a garantir a qualidade, confiabilidade e baixo custo de produção de tais produtos.

Nesta seção de revisão bibliográfica são apresentados métodos tradicionais e métodos que fazem uso da IA (Inteligência Artificial), sendo que eles podem ser aplicados na análise de trilhas e componentes seja com o intuito de encontrar defeitos, anomalias ou segmentar e classificar os componentes da PCB. Os trabalhos citados serão divididos em 3 categorias: registro da PCB, métodos tradicionais e métodos que fazem o uso da IA. Ainda dentro dos métodos que utilizam IA, existe uma categorial especial que são os métodos que utilizam o DL (*Deep Learning*).

3.1 Registro da PCB

No registro da PCB são realizadas transformações na imagem, de modo que seja possível eliminar rotações, translações e segmentar a PCB do fundo da imagem. No caso do presente trabalho, o fundo da imagem é representado pela esteira de transporte.

O uso de *features*, no registro de uma imagem, é uma técnica comumente utilizada. Os algoritmos SURF (*Speeded-Up Robust Features*) e SIFT (*Scale-Invariant Feature Transform*) servem de base à implementação de novos algoritmos de extração de *features*, conforme será citado nos trabalhos apresentados nesta seção.

Em Dai et al.[53] é proposto um novo algoritmo para o registro batizado de SIFT-PSO otimizado. O algoritmo proposto diminui o tempo de processamento e melhora a eficiência do registro da PCB. A Tabela *2* mostra o RMSE (*Root Mean Square Error*) considerando 30 imagens de PCB.

Tabela 2 - Comparação entre métodos de extração de *features*, Adaptado de *[53]*

Método	RMSE	Processamento(s)
SIFT	0,7263	29,8050
SURF	0,9497	0,8900
SIFT-PSO	0,5652	19,3240
SIFT-PSO Otimizado	0,5146	10,5360

Em Hua et al. [54] as *features* são encontradas pelo algoritmo GFTT (*Good Features To Track*). Assim, apenas a parte de cálculo dos descritores proposto pelo SURF é usado, enquanto a correspondência entre as *features* da imagem de referência e da imagem em análise são calculados pela distância Euclidiana. Então, utiliza-se o RANSAC para calcular os parâmetros necessários à transformação que resulta no registro da imagem. Após o primeiro registro, a correlação cruzada é utilizada para verificar a translação entre a imagem em análise e a imagem referência proporcionando um registro de melhor qualidade. Segundo os dados apresentados pelo artigo, o RMSE calculado após o registro da imagem é inferior a 0,1000.

3.1.1 Métodos Tradicionais

Em relação aos métodos tradicionais destinadas a detectar defeitos em PCB, em geral eles utilizam as seguintes operações de processamento de imagem: subtração de imagens, comparação de imagens com um padrão pré-determinado, aplicação de operações morfológicas e análise de *features* de elementos conectados em imagens binárias (*Blob – Binary Large Objects*). Os trabalhos apresentados a seguir seguem essa linha de raciocínio.

Zhu et al.[55] propuseram um método de análise de falhas em PCB capaz de detectar 5 tipos de defeitos: curto-circuito, circuito aberto, depressão, protusão e furos. Este método trabalha com imagens em tons de cinza, submetidas posteriormente para remoção de ruídos utilizando as técnicas de *Wavelet denoising* [56] e equalização de histograma. Posteriormente a esta etapa, as imagens são binarizadas utilizando o método de Otsu [19], seguido pela diferença entre uma imagem padrão da PCB e a imagem em análise. Como para realizar a diferença

entre as imagens é necessário que as imagens em teste sejam registradas, Zhu et al. utiliza a transformação de *Hough* [19] para esse fim. Finalmente são aplicadas operações morfológicas de fechamento, visando remover pequenas descontinuidades nas bordas dos elementos da PCB, além da remoção de ruídos através da aplicação da operação morfológica de abertura com um elemento estruturante circular. Ao final desse processo é possível encontrar os defeitos, sendo que a classificação é feita analisando-se as seguintes características dos *Blobs*: número de conexões e área do defeito. Considerando 30 imagens de PCB defeituosas o algoritmo proposto por Zhu et al. conseguiu encontrar 100% dos defeitos, classificando corretamente 90% deles.

Tsai e Huang [21] propuseram um método de identificação de defeitos em PCB baseado na comparação da placa em teste com uma placa padrão (*template)*. Nas metodologias que propõem o uso da correspondência de modelos (*template matching*), geralmente utiliza-se a diferença entre a placa-modelo e a placa em teste, classificando a placa em teste a partir de um limiar de aprovação. Porém, esse método requer um alinhamento preciso entre a placa-modelo e a placa em teste, sendo também susceptível à iluminação e variações randômicas do produto em teste. Para resolver esse problema os autores calculam a 2D DFT (*2D Discrete Fourier Transform*) das imagens de teste e do modelo. A partir da diferença entre os espectros reconstrói-se uma nova imagem, eliminando-se os padrões comuns e retendo apenas as possíveis anomalias. Finalmente é aplicado um limiar (*threshold)* adaptativo para encontrar os defeitos. A vantagem desse método é que não é requerida nenhuma espécie de treinamento, sendo também invariante à translação, rotação e iluminação. Testes com um conjunto de 40 imagens sem defeitos e 55 imagens com defeito, para PCBs de diversos produtos eletrônicos, mostraram uma taxa de detecção de 100% considerando os seguintes defeitos: arranhões, circuitos abertos, curtos-circuitos, ausência e desalinhamento de componentes.

Em Hassanin et al.[4] é proposto um método que utiliza-se da extração e seleção de *features* para encontrar a PCB e eliminar a rotação. Além disso, as *features* também são utilizadas para detectar a ausência de componentes. O método SURF, que pode ser visto como uma implementação mais eficiente do algoritmo SIFT, é utilizado no referido artigo para extração e seleção de *features*. Segundo o autor, a grande contribuição do trabalho está no fato de que o algoritmo proposto consegue identificar e determinar qualquer tipo de defeito em qualquer PCB, não

importando a orientação e posição desta. Porém, somente 13 imagens foram utilizadas no teste, onde foram encontrados os seguintes tipos de defeitos: componentes ausentes, trilhas abertas e letras faltantes na serigrafia das PCBs.

Chaudhary et al.[10] propuseram um método de classificação de PCB defeituosas que pode ser dividido em 4 estágios: registro da imagem, pré-processamento, segmentação e classificação dos defeitos. O registro da imagem é feito utilizando-se a extração de *features* por meio do algoritmo FAST (*Features from Accelerated Segment Test*), seguido pela correspondência de *features* entre a PCB em análise e a referência com o algoritmo SSD (*Sum of Squared Difference*) e, finalmente, pela transformação geométrica calculada pelo algoritmo MSAC (*M-estimator Sample Consensus*). No pré-processamento tenta-se remover os ruídos e intensificar os detalhes da imagem por meio da aplicação de filtros de mediana (7x7) e de filtros passa-baixas Gaussianos. Na segmentação são aplicadas as técnicas de limiar por histograma e operações morfológicas. Neste trabalho são segmentadas as trilhas e os *pads* de solda. Finalmente, a identificação dos defeitos é feita pela subtração de imagem. Como resultado, o trabalho conseguiu identificar 14 tipos de defeitos relativos as trilhas e pads, em aproximadamente 2,53 s.

Em Annaby et al.[57] é proposto um algoritmo para detectar componentes ausentes em PCBs. O método proposto foi batizado com o nome INCC (*Improved Normalized Cross-Correlation*), apresentando melhorias em relação ao NCC (*Normalized Cross-Correlation*), como robustez contra ruído e tolerância a variações de iluminação e brilho. O INCC é baseado na comparação entre uma imagem padrão e uma imagem em teste. Primeiramente, as duas imagens são convertidas para um vetor 1D e é criado um vetor de *features* após a aplicação da transformada discreta dos cossenos (DCT – *Discrete Cosine Transform*)[58] no vetor 1D. Por fim, calcula-se a NCC entre os dois vetores de *features* e, através de um limiar, classifica-se a imagem como defeituosa ou não. Considerando um conjunto de 32 pares (imagem sem defeito e imagem defeituosa), distribuídas entre imagens de PCBs de *PC desktops* e PCBs de telefones móveis, chegou-se a uma acurácia de aproximadamente 96,00%.

3.2 Métodos que fazem o uso da Inteligência Artificial (IA)

3.2.1 Métodos Tradicionais

Nesta seção da revisão bibliográfica são apresentados métodos que não fazem uso do DL. Aqui os autores utilizaram algoritmos clássicos, como o *SVM* (*Support Vector Machine*) *e Random Forest*.

Qinqin He et al.[59] propuseram um *framework* baseado em ML (*Machine Learning*) para a detecção de defeitos em PCB, em que tais defeitos estão relacionados especificamente a trilhas e *pads*. O framework *é* baseado na extração de *features* através dos métodos HOG (*Histogram of Oriented Gradients*) e LBP (*Local Binary Pattern*). O conjunto de *features* provenientes de ambos os métodos é utilizado para treinar dois modelos SVM, que posteriormente são combinados utilizando a fusão Bayesiana. O conjunto de imagens de PCB, batizado de PCBSET, é formado por blocos das imagens originais. Inicialmente, as imagens de PCB são segmentadas e passam pela remoção de ruídos através de filtros de mediana e média, sendo posteriormente divididas em blocos de 256 x 256 pixels, originando o conjunto PCBSET que contém 148 blocos defeituosos e 148 blocos livres de defeitos. Deste conjunto, 56 imagens de cada bloco foram utilizadas no teste, resultando em uma acurácia de 89,22%. A inovação defendida pelos autores é que o método de fusão de *features,* provenientes do HOG e LBP, apresenta uma maior acurácia em relação à adoção das *features* oriundas de um único método (HOG ou LBP). No artigo são apresentadas a acurácia para o uso de somente o LBP (85,71%) e o HOG (87,50%) justificando assim o ganho representado pela fusão dos modelos.

Em Oliveira et al.[60] é apresentado um sistema de inspeção visual projetado para detectar fraudes nas PCB de controle de bombas de combustível. O SIFT é utilizado para extrair as *features* usadas no registro da imagem e na classificação das placas. Nesse trabalho o registro é feito usando a homografia a partir da correlação de *features*, estas provenientes da imagem em análise e de uma imagem referência, encontradas pelo método RANSAC. Na classificação das PCB em fraudadas ou não, o algoritmo SVM é utilizado, e o estudo chegou a uma precisão de 77,00% considerando 572 imagens de placas não adulteradas e 77 imagens de placas adulteradas.

Em Li et al.[61] é proposto um algoritmo para inspecionar a correta inserção dos componentes da PCB. O método proposto utiliza um classificador *Random Forest,* baseado na profundidade das imagens (*Depth Images*) para segmentar e classificar os componentes. Foram utilizadas 200 imagens sintéticas para o treinamento, enquanto na avaliação do modelo foram utilizadas 40 imagens sintéticas e 10 imagens reais. O algoritmo proposto alcançou uma acurácia média de 98,96% nas imagens sintéticas e 83,64% nas imagens reais, e o tempo de processamento para classificar os componentes da PCB (32 componentes) foi de 0,9 s.

Yuk et al.[3] propôs um método não-referencial, baseado em IA, capaz de detectar arranhões em PCB. Tais defeitos, podem ocasionar mal funcionamento do produto através de curtos-circuitos ou circuitos abertos. Inicialmente, o método proposto utiliza o algoritmo SURF para a extração das *features* nas PCB. Esse algoritmo foi escolhido pois é invariante às dimensões da imagem, à rotação e à localização da PCB na imagem capturada. As áreas com as falhas são identificadas manualmente, possibilitando a classificação dos vetores de *features* em vetores que representam falhas e vetores livre de falhas. Posteriormente, as *features* respectivamente categorizadas são submetidas ao método de classificação *Random Forest*, sendo este escolhido por ser rápido ao classificar dados de alta dimensionalidade e ser menos susceptível a problemas de *overfitting* [14], [15] (graças ao conjunto de diversas árvores de decisão). As probabilidades calculadas pelo passo anterior irão formar um mapa WKDE (*Weighted Kernel Density Estimator*), cujas áreas que representam falhas irão apresentar um valor maior que um determinado limiar. No treinamento e posterior teste utilizou-se 10 imagens com arranhões, cuja a métrica utilizada na análise foi a AUROC (*Area Under ROC Curve*), apresentando o valor 0,91 para o método descrito.

3.2.2 Métodos com Redes Neurais

Nesta seção são apresentados os trabalhos que fazem uso de CNN. Além disso, alguns métodos fazem uso do TL (*Transfer Learning)* e do treinamento não supervisionado.

Adibhatla et al.[62] desenvolveram um sistema para a detecção de defeitos em PCB utilizando a arquitetura CNN Tiny-YOLO-v2. As imagens utilizadas no estudo

foram coletadas a partir de máquinas AOI e convertidas no formato JPEG. Uma interface gráfica foi criada, permitindo aos engenheiros de qualidade pré-definir as áreas defeituosas e criar classes de defeitos. Inicialmente foram definidos 11 tipos de defeitos, e embora eles não tenham sido detalhados, o trabalho deixa evidente que qualquer classe de defeito pode ser definida. Um total de 11000 imagens foram utilizadas no treinamento e validação, com tamanhos de *batch* 8, 16 e 32, enquanto os hiper parâmetros foram encontrados por meio do método de tentativa e erro. Assim, considerando o *batch* de 32, conseguiu-se uma acurácia de 98,82%. Dado o tamanho relativamente pequeno dos pesos da rede neural, sua arquitetura tem a vantagem de poder ser executada em sistemas embarcados de relativo baixo custo.

Volkau et al.[63] propuseram um método para identificação de defeitos em placas PCB baseado no TL para a arquitetura da rede neural VGG16 pré-treinada com as imagens do conjunto ImageNet. Embora a VGG16 seja geralmente utilizada na classificação com treinamento supervisionado, dada a dificuldade de se obter milhares de imagens de forma balanceada entre imagens defeituosas e imagens sem defeito, os autores propuseram uma forma de treinamento não-supervisionado, utilizando algumas dezenas de imagens sem defeito. No modelo proposto, as camadas convolucionais da VGG16 pré-treinadas são congeladas e as camadas totalmente conectadas são treinadas considerando rotações das imagens (0º, 90º, 180º e 270º), com cada imagem dividida em 66 partes representando as classes de treinamento. Após o treinamento, a penúltima camada conectada e a camada *softmax* são removidas, e o vetor de *features* resultante da camada totalmente conectada restante é utilizado na classificação. Assim, cria-se, com as imagens sem defeito, um agrupamento inicial. Logo, se a distância entre o vetor de características relativo a uma imagem de teste for maior que um determinado limiar, essa amostra é classificada como defeituosa. Três conjuntos de imagens foram utilizadas no treinamento e validação: PCB-G (48 imagens sem defeito e 200 imagens com defeitos), PCB-1(78 imagens sem defeito e 155 imagens defeituosas) e DS (1814 imagens defeituosas e 2251 imagem sem defeitos). Os dois primeiros conjuntos referem-se a arranhões, furos e cantos defeituosos, já no segundo conjunto tem-se imagens de defeitos em solda. A acurácia nos 3 conjuntos foi em média de 90%.

No trabalho apresentado por Mallaiyan Sathiaseelan et al.[64] inicialmente são discutidos os desafios de se utilizar IA e redes neurais para a detecção e classificação de componentes em PCB. O artigo também lista possíveis aplicações

para o assunto abordado, citando, inclusive, a detecção de *trojans* de *hardware*, além das aplicações mais fundamentais, como detecção de defeitos. Finalmente, o artigo propõe um método para extração da BOM (*Bill Of Materials)* de uma PCB, método esse batizado de ECLAD Net (*Eletronic Component Localization and Detection-Network*). Durante a descrição do método utilizado, os autores citam os desafios no uso de modelos genéricos baseados em DL: grande variância intra-classes, pequena variância inter-classes, mudança constante no projeto das placas e falta de banco de imagens. O modelo proposto é um comitê de métodos tradicionais e redes neurais. Embora a proposta seja identificar toda a lista de materiais em qualquer PCB, os resultados apresentados são baseados na identificação de resistores e capacitores em 26 imagens de PCBs, alcançando assim uma acurácia de 98,90%.

A detecção de anomalias, usando métodos de aprendizado não-supervisionado, pode ser aplicado em diversos setores da indústria, principalmente na indústria eletrônica, conforme apresentado no trabalho de Wang et al[65]. Neste trabalho é proposto uma UAD-GAN (*Unsupervised Anomaly Detection GAN*) que faz o uso do TL baseado na Inception-V3. Desse modo, as imagens normais são submetidas à *Inception V3* e as *features* extraídas da penúltima camada são usadas para treinar o discriminador e o gerador da GAN. O gerador irá aprender por meio da distribuição de *features* extraídas pela *Inception* e o discriminador será capaz de reconhecer se essa distribuição pertence às imagens sem anomalias. O conjunto de imagens utilizado no trabalho foram capturadas por uma máquina de SPI (*Solder Past Machine)*, um total de 4430 imagens foram usadas no treinamento e 1525 imagens foram usadas na validação, destas 417 imagens apresentavam anomalias. As imagens referenciam-se a um determinado *pad*, região onde os componentes serão soldados na PCB. Considerando esse conjunto de dados, o método apresentou um AUC (*Area Under Curve*) de 0,8067. Além disso, 6,23% das peças defeituosas não foram detectadas e o índice de alarmes falsos foi de 74,83%.

Segundo Li et al.[1], a detecção de componentes eletrônicos em placas PCB é essencial na garantia da qualidade dos produtos e fundamental na montagem assistida por robôs. Sendo que essa montagem inteligente é fundamental na era da indústria 4.0. A montagem e inspeção de PCB nas indústrias eletrônicas 3C (*Computer, Communication and Consumer Electronics*) fazem parte desses trabalhos repetitivos, ou seja, a inspeção e montagem automática de PCB fazem

parte da tendência de transformação desse ramo industrial. Tanto na montagem quanto na inspeção de PCB, a detecção dos diversos componentes e suas devidas localizações, são essenciais ao processo de automação. O trabalho apresentado em [1] propõe um modelo *DL* que retorna a posição (*bounding box)* e a categoria dos componentes baseado na rede neural YOLO. O trabalho faz uma mudança na arquitetura da rede adicionando uma camada de saída ao YOLO e gerando 12 *anchor boxes* para a detecção dos componentes eletrônicos. O conjunto de treinamento foi composto por 47 imagens do conjunto *pcb_wacv_2019,* além de imagens sintéticas providas pelas bibliotecas do software *Altium Designer.* Por outro lado, o conjunto de testes é formado por 50 imagens de PCB encontradas na internet. O total de componentes presentes nas 50 imagens é de 9145 componentes, representando 30 categorias, tais como: resistores, capacitores, transistores, diodos, botões, dissipadores de calor, circuitos integrados, etc. Esse conjunto de teste também foi submetido a um processo de DA (*Data Augmentation*)*,* aumentando o seu tamanho em 20 vezes. Com esse conjunto, chegou-se a um mAP (*mean average precision)* de 93,07% contra 77,08% considerando o YOLO V3 sem as modificações propostas.

Em Kuo et al.[66] é proposto um método baseado em DL com o objetivo de detectar e classificar componentes (resistores, capacitores, led, indutores, etc) em PCBs. O propósito final da classificação e detecção de componentes, segundo o artigo, seria proporcionar a inspeção automática das PCBs e o processo de engenharia reversa de *hardware*. Além disso, esse processo de identificação dos componentes seria importante para fornecer bases de dados anotadas para outros métodos de DL, uma vez que a anotação manual exige profissionais treinados, sendo um processo demorado e, portanto, custoso. Além de citar a falta de dados relativos à PCB, o trabalho elenca como uma das principais dificuldades a de se criar métodos de detecção de componentes, a pouca variação entre classes e a alta variação intraclasse. Como exemplos do primeiro caso pode-se citar resistores e capacitores, enquanto exemplos para o segundo caso são os conectores que apresentam diversos formatos. O método desenvolvido é composto por 3 etapas. Inicialmente as imagens são apresentadas a uma RPN (*Region Proposal Network*), sendo esta responsável por predizer as regiões com potencial para representar componentes. As *features* de tais regiões são então submetidas a um GN (*Graph Network*), sendo esta responsável por refinar as *features* levando em consideração

elementos estruturais (nós e bordas). Por fim, as *features* são submetidas a uma SPN (*Similarity Prediction Network*) que irá verificar a similaridade em relação a uma base de componentes *template.* O conjunto de dados utilizado no experimento foi formado por 48 imagens anotadas por *experts* em PCB, onde 40 foram utilizadas para treinamento e 8 imagens foram utilizadas para teste. Como resultado, o método descrito obteve uma acurácia de 82,00% e um mAP de 0,6530. A principal contribuição, segundo os autores, foi a modificação proposta na GN e o uso do método composto pelas 3 etapas (RPN, GN e SPN), conseguindo alcançar uma acurácia significativa mesmo utilizando poucas amostras no processo de aprendizado.

Em Zhang et al.[67] é proposta uma CNN capaz de classificar 6 tipos de falhas: curto-circuito, circuito aberto, *spurius copper, mousebit e spur*. A arquitetura da rede é composta por 3 estágios, em que cada um deles é formado por uma camada convolucional, uma função de ativação (*ReLU*) e uma camada de *pooling (max pooling)*. Após os 3 estágios existe uma camada conectada onde ocorre a classificação. O diferencial nesse trabalho é que uma imagem pode ser classificada em mais de uma categoria, dependendo dos erros detectados. Considerando 640 imagens de teste (50% com defeitos e 50% sem defeitos), a acurácia média, dada a classificação entre as 6 categorias foi de 89,89%, enquanto a acurácia considerando a classificação em duas categorias (PCB defeituosa ou não) foi de 92,86%.

Em Lim et al.[68], é utilizada a segmentação semântica, mais especificamente uma rede FCN (*Fully Convolutional Network*), para segmentar os elementos a serem analisados: capacitores, indutores, cristais de oscilação e CI (Circuitos Integrados). Com as regiões dos componentes delimitadas, são aplicadas operações morfológicas e operações de intersecção com uma imagem de referência. Após esse pré-processamento, uma segunda CNN (*Convolutional Neural Network*) é aplicada de forma a classificar o componente. O intuito do trabalho não é encontrar defeitos e sim especificar áreas da PCB a serem analisadas por outros sistemas de AOI. Considerando um conjunto de treinamento de 7659 imagens e um conjunto de teste de 4822 imagens, chegou-se a uma acurácia média de 90,8%.

Em Shi et al.[11] um método de aprendizado semi-supervisionado é proposto para encontrar defeitos em PCB relacionados às trilhas: curto-circuito, circuito aberto, *spurius copper, mousebit e spur*. O método apresentado nesse trabalho é uma adaptação em relação às GANs onde a combinação de 3 sub-redes foi utilizada

para detectar anomalias em PCB: gerador, discriminador e codificador. A avaliação das anomalias é feita pela diferença dos vetores de características provenientes da imagem real e da imagem gerada. Considerando um conjunto de 10 imagens sem defeitos utilizadas no treino e um conjunto de 120 imagens defeituosas utilizadas no teste, o método proposto alcançou um AUC de 0,869.

Em Shi et al.[69] é apresentado um método, baseado em CNN capaz de detectar pequenos defeitos em PCB. O método foi chamado de SSDT (*Single Shot object DeTector*) e, assim como em alguns trabalhos já apresentados, é capaz de detectar defeitos em trilhas. Considerando o conjunto de dados PASCAL VOC o algoritmo proposto superou, em termos de acurácia, os métodos apresentados na Tabela 3.

Tabela 3 - Acurácia SSDT. Fonte [69]

Algoritmo	**Porcentagem de acurácia que o SSDT foi superior aos algoritmos da primeira coluna da tabela.**
Fast R-CNN	9,1 %
Faster RCNN com backbone VGG	5,9%
Faster RCNN com backbone ResNet	2,7%

Mesmo com o uso do DA, as CNNs requerem uma grande quantidade de imagens para o treino, principalmente para a classificação entre peças boas e defeituosas. Nas aplicações reais da indústria eletrônica não existe essa quantidade de amostras de imagens defeituosas, assim em Ke et al. [6] é proposto um método não supervisionado para a avaliação de anomalias. O trabalho detalhado em [6] não é relacionado à PCB, porém apresenta semelhanças com o proposto neste trabalho. Em [6] é proposto um CAE capaz de identificar defeitos na serigrafia em *smartphones* (logo da empresa e modelo do aparelho). O CAE consegue extrair as características efetivas do objeto em análise utilizando somente as imagens sem defeito, não havendo a necessidade de atribuição de classes. Deste modo, a rede aprende a criar um *template* do objeto sendo possível detectar as anomalias fazendo a diferença entre o *template* e a imagem em teste. O trabalho conseguiu uma

acurácia média de 98,90% em 1693 imagens de teste de 3 modelos diferentes de *smartphones*.

Em Khalilian et al.[70] também é proposto o uso de um *Autoencoder*, nesse caso um *Denoising Convolutional Autoencoder*, capaz de detectar e localizar defeitos em trilhas de PCB. O algoritmo proposto no trabalho também é capaz de reconstruir a parte danificada no circuito, salientando em detalhes o tipo de falha. A base de dados utilizada consiste em 1500 imagens de PCB defeituosas, sendo que tais imagens estão alinhadas com imagens sem defeito. Considerando esse conjunto o trabalho alcançou uma acurácia de 97,50%.

Em Xia et al.[71] é apresentada uma junção entre um método tradicional e uma rede neural, cujo algoritmo recebeu o nome de SSIM-NET, método esse utilizado para avaliar defeitos em trilhas e soldas. O SSIM (*Structural Similarity Index*) [72] é utilizado para encontrar regiões onde existe a possibilidade de ocorrência de defeito. O simples uso do SSIM nesta detecção é susceptível a variação da iluminação, ruído da câmera, etc. Assim, os autores utilizam a rede Neural *MobileNet-V3* para classificar as regiões previamente selecionadas pelo SSIM. O artigo não deixa claro quantas imagens são utilizadas no treino da *MobileNet-V3*, porém apresenta um resultado mAP de 97,06% considerando um banco de dados de 1386 PCB's.

Em Khare et al.[73] é uma apresentado um método, baseado na rede *YOLOv3,* capaz de detectar componentes ausentes ou componentes inseridos incorretamente na PCB. O conjunto de imagens utilizados no trabalho é composto por 530 imagens de PCBs de placas mães de computadores. No treinamento da rede neural foram utilizadas 459 imagens, enquanto que 51 imagens foram utilizadas na validação, e as 20 imagens restantes foram utilizadas no teste. Os componentes analisados capacitores, indutores, resistores e CI foram anotados manualmente através da ferramenta *labelImg*. Como resultado obteve-se uma acurácia de 75,48%.

Em Wu et al.[74] são comparados os resultados de duas redes neurais, responsáveis por localizar e classificar defeitos em trilhas e sodas de PCBs. A primeira rede neural testada foi a rede SSDT, que se utiliza da rede VGG16 como *backbone* atrelada a uma camada de predição e subsequente filtragem de resultados, executada pelo algoritmo NMS (*Non-Maximum Suppression*). A segunda rede utilizada na comparação é a FPN (*Feature Pyramid Network*). Neste caso, a rede *resnet101* foi utilizada como *backbone.* Nos testes, foram utilizados dois conjuntos de dados: PCB *Dataset* (1386 imagens) e DeepPCB *Dataset* (1500 pares

de imagens, registradas). O artigo não deixa evidente o processo de treinamento utilizado, apresentando como o melhor resultado a acurácia alcançada pela rede FPN 96,40% no PCB *Dataset* e 97,30% no DeepPCB *Dataset*.

Em Silva et al.[75] duas redes neurais VGG16 e ResNet150 são utilizadas na detecção de defeitos em PCBs. O banco de imagens DeepPCB é novamente utilizado nesse estudo. Este banco de imagens contém 1500 pares de imagens (*template* e imagens de teste), sendo que as imagens de teste possuem todos os defeitos previamente anotados. O conceito de TL é aplicado nesse trabalho, sendo assim, durante o processo de *fine-tuning* foram executados diversos testes congelando diferentes camadas. Para o treinamento e validação de ambas as redes foram utilizadas 1000 imagens, restando 500 imagens para o teste. A ResNet150 não conseguiu apresentar um resultado satisfatório com nenhuma das combinações de congelamento de camadas no processo de *fine-tuning.* Já a rede Vgg16 alcançou uma acurácia de 89% considerando a remoção de 4 camadas finais responsáveis pela extração de features, e treinamento das próximas 4 camadas subsequentes às removidas utilizando o conjunto DeepPCB.

Em Li et al. [76] é proposto um comitê de redes neurais com o objetivo de inspecionar regiões de solda em PCBs. A primeira rede neural utilizada no estudo foi a rede FRRF: combinação das redes Faster RCNN (*Regional Based Convolutional Neural Network*) e FPN, sendo utilizada a rede ResNet-101 como extrator de *features*. Essa rede alcançou uma acurácia de 98,40% na detecção de falhas. A segunda rede avaliada no estudo foi a YOLOv2, sendo que esta conseguiu uma acurácia de 97,89% da detecção de falhas. O comitê com ambos os métodos alcançou uma acurácia de 96,73%, porém diminuiu a taxa de falsos alarmes de 50% para 19,73%. No treinamento de ambas as redes foram utilizadas 834 imagens de PCBs, um processo de aumento da base de dados também foi utilizado para balancear as classes de defeitos.

Em Xie et al. [77] é utilizada uma rede neural baseada no YOLOv4 para detectar falhas em trilhas de PCBs. As seguintes melhorias são propostas no artigo: uso do MRHAM (*Multistage Residual Hybrid Attention Module*) para a extração de *features*, uso do *K-means++* para melhorar a acurácia na detecção de pequenos objetos, uso do DA e TL no processo de treinamento. Considerando um conjunto de 693 imagens conseguiu-se um mAP de 99,71%.

3.2.3 Resumo dos trabalhos apresentados na revisão.

As Tabelas 4, 5 e 6 trazem um resumo dos artigos apresentados na revisão bibliográfica. O objetivo de tal resumo é evidenciar ao leitor a variedade de trabalhos relacionados à inspeção de PCBs. Além disso, fica também evidente a variedade de métodos, tipos de PCB, aplicações da inspeção e métricas utilizadas para avaliar os métodos. Uma vez que não é simples estabelecer o melhor método geral para ser utilizado na inspeção visual, possivelmente a melhor escolha é a combinação de métodos em um comitê.

Considerando o fato de que na indústria eletrônica, dada a existência de outros métodos para encontrar peças defeituosos como por exemplo o ICT (*In Circuit Test*), é muito difícil construir uma base de imagens com defeitos. Além disso, o processo de marcação das áreas defeituosas é um processo demorado exigindo mão de obra qualificada. Portanto, dentre todos os métodos de DL, aqueles baseados em aprendizado não supervisionado, que fazem o uso do TL, parecem ser os mais indicadas para comporem o comitê.

Finalmente, todos os trabalhos de alguma forma fazem o registro da imagem em teste. Esse registro pode ser proporcionado por meio de transformações na imagem ou mesmo fisicamente, fazendo o uso de algum dispositivo mecânico.

Tabela 4 - Resumos artigos que fazem uso de métodos tradicionais.

Métodos Tradicionais.					
#	**Autor(es)**	**Métodos**	**Aplicação**	**Quantidade de Imagens**	**Resultados**
1	Zhu et al.[55]	*Wavelet denoising,* Otsu, *Hough.*	Curto-circuito, circuito aberto, depressão, protusão e furos.	30	Acurácia de 90% na classificação de defeitos.
2	Tsai e Huang [21]	Reconstrução pela transformada de Fourier bidimensional.	Arranhões, circuitos abertos, curtos-circuitos, ausência e desalinhamento de componentes.	40 imagens sem defeito e 55 imagens defeituosas.	Acurácia de 100% na detecção de defeitos.

3	Hassanin et al.[4]	SURF, extração de *features*.	Componentes ausentes, trilhas abertas e letras faltantes na serigrafia da PCBs.	13	Detecção de qualquer tipo de defeito em PCB.
4	Chaudhary et al.[10]	FAST, SSD, MSAC, segmentação.	Trilhas e os *pads* de solda.	Não especificado no trabalho.	14 tipos de defeitos em 2,53 s.
5	Annaby et al.[57]	INCC, DCT	Componentes Ausentes.	32 pares de imagens.	Acurácia de 96%.

Tabela 5 – Resumo dos artigos que utilizam métodos tradicionais de IA.

Métodos Tradicionais de IA.					
#	**Autor(es)**	**Métodos**	**Aplicação**	**Quantidade de Imagens**	**Resultados**
1	Qinqin He et al.[59]	HOG, LPB, extração de *features*, SVM, teoria Bayesiana.	Trilhas e pads.	56	Acurácia de 89,22%.
2	Oliveira et al.[60]	SIFT, RANSAC, SVM.	Alterações em PCBs de bombas de combustível.	572 imagens de PCBs não adulteradas e 77 imagens de PCBs adulteradas.	Acurácia de 77% na detecção das adulterações.
3	Li et al.[61]	*Random Forest*, profundidade de Imagens.	Classificação de 32 componentes.	200 imagens sintéticas no treinamento, 40 imagens sintéticas e 10 imagens reais no teste.	Acurácia de 98,96% nas imagens sintéticas e de 83,64% nas imagens reais.
4	Yuk et al.[3]	SURF*, Random Forest*, WKDE.	Arranhões.	10 imagens para treinamento e teste.	AUROC de 0,91.
5	Annaby et al.[57]	INCC, DCT	Componentes Ausentes.	32 pares de imagens.	Acurácia de 96%.

Tabela 6 - Técnicas de IA que fazem uso de redes Neurais.

Métodos de IA com redes neurais.					
#	**Autor(es)**	**Métodos**	**Aplicação**	**Quantidade de Imagens**	**Resultados**
1	Adibhatla et al.[62]	Tiny-YOLO-v2	Qualquer tipo de defeito em PCBs	11000	Acurácia de 98,82%.
2	Volkau et al.[63]	VGG16, ImageNet, *Transfer Learning*	Arranhões, furos ausentes, cantos quebrados e problemas em soldas.	3 Conjuntos: A(1): 48;200. B(2): 78;155. C(3):1814;2251.	Acurácia de 90%.
3	Mallaiyan Sathiaseelan et al.[64]	Comitê, Técnicas Tradicionais e Redes Neurais.	Identificação de resistores e capacitores	26	Acurácia de 98,9%.
4	Wang et al[65]	UAD-GAN, *Inception V3, Transfer Learning*.	Problemas em soldas relativos ao determinado *pad*.	4430: Treinamento. 1525: Validação.	AUC de 0,8067.
5	Li et al.[1]	YOLO Modificado.	Identificação de components.	47: Treinamento. 50: Teste.	mAP de 93,07%.
6	Kuo et al.[66]	RPN, GN, SPN.	Identificação de componentes.	40: Treinamento. 8: Teste	Acurácia de 82% e mAP de 0,653.
7	Zhang et al.[67]	CNN, ReLu.	Falhas em trilhas: curto circuito, circuito aberto, *spurius copper*, *mousebit* e *spur*.	640	Acurácia de 92,86%.
8	Lim et al.[68]	FCN, AND, CNN.	Segmentação de componentes: capacitores, indutores, CI.	7659: Treinamento. 4822: Validação.	Acurácia de 90,8%.

9	Shi et al. [11]	GAN	Falhas em trilhas: curto circuito, circuito aberto, *spurius copper*, *mousebit* e *spur*.	10: Treinamento. 120: Teste.	AUC de 0,869.
10	Shi et al.[69]	SSDT, CNN	Detectar pequenos defeitos em trilhas.	Conjunto PASCAL VOC(4).	Acurácia superior aos algoritmos semelhantes em até 9,1%.
11	Em Khalilian et al.[70]	*Denoising Convolutional Autoencoder*.	Falhas em Trilhas.	3000 imagens com e sem defeito.	Acurácia de 97,5%.
12	Xia et al.[71]	SSIM-NET, MobileNet-V3.	Falhas em trilhas e soldas.	1386	mAP de 97,6%.
13	Khare et al.[73]	YOLOv3	Detectar componentes ausentes.	459: Treinamento. 51: Validação. 20: Teste.	Acurácia de 75,48%.
14	Wu et al.[74]	SSD, VGG16, NMS, FPN.	Falhas em trilhas e soldas.	1500 pares de imagens.	Acurácia de 97,3%.
15	Silva et al.[75]	VGG16, ResNet150, *Transfer Learning, Fine Tuning*.	Falhas em trilhas e soldas.	1000: treinamento. 500: Teste.	Acurácia de 89%.
16	Li et al. [76]	*Faster* RCNN, YOLOv2, FPN, Comitê.	Falhas em regiões de Solda.	834.	Acurácia de 96,73%.
17	Xie et al. [77]	YOLOv4, MRHAM, *Kmeans++, Transfer Learning*.	Falhas em trilhas.	693.	mAP de 99,71%.

18	Presente Trabalho	U-Net, *Watershed*, 2D DFT, *Autoencoder*, VGG, GAN, Comitê.	Ausência e defeitos em componentes de PCB de *smartphones*.	Modelo A: 17 imagem sem defeito e 18 imagens com Defeito. Modelo B: 54 imagens sem defeito.	*IoU Score* médio de 0,9835. Acurácia média de 90,87%.

(1)A: Conjunto de imagens PCB-G (418 imagens sem defeito e 200 imagens com defeitos).

(2)B: Conjunto de imagens PCB-1 (78 imagens sem defeito e 155 imagens com defeitos).

(3)C: Conjunto de imagens DS (1814 imagens com defeito e 2251 imagens sem defeito).

(4)PASCAL VOC (*Visual Object Classes Challenge*): conjunto de dados utilizado em competições de classificação e detecção.

4 Metodologia

Neste capítulo, inicialmente, é apresentado o *hardware* utilizado para a aquisição das imagens utilizadas no estudo. Após essa discussão, são apresentados os detalhes da inspeção visual aplicada aos componentes de dois modelos de *smartphones*, por fim são detalhados os métodos que compõem o comitê proposto.

4.1 Aquisição das Imagens

A Figura 23 exibe os componentes de *hardware* para a aquisição de imagens. Os *brackets* estão dispostos na esteira de produção, onde é garantido apenas o espaçamento mínimo entre as peças, não existindo nenhum sistema para a indexação (sistema mecânico que orienta o *smartphone* removendo possíveis rotações).

A Tabela 7 apresenta as características do sistema de visão, necessárias para aquisição e avaliação dos *brackets* considerados neste trabalho.

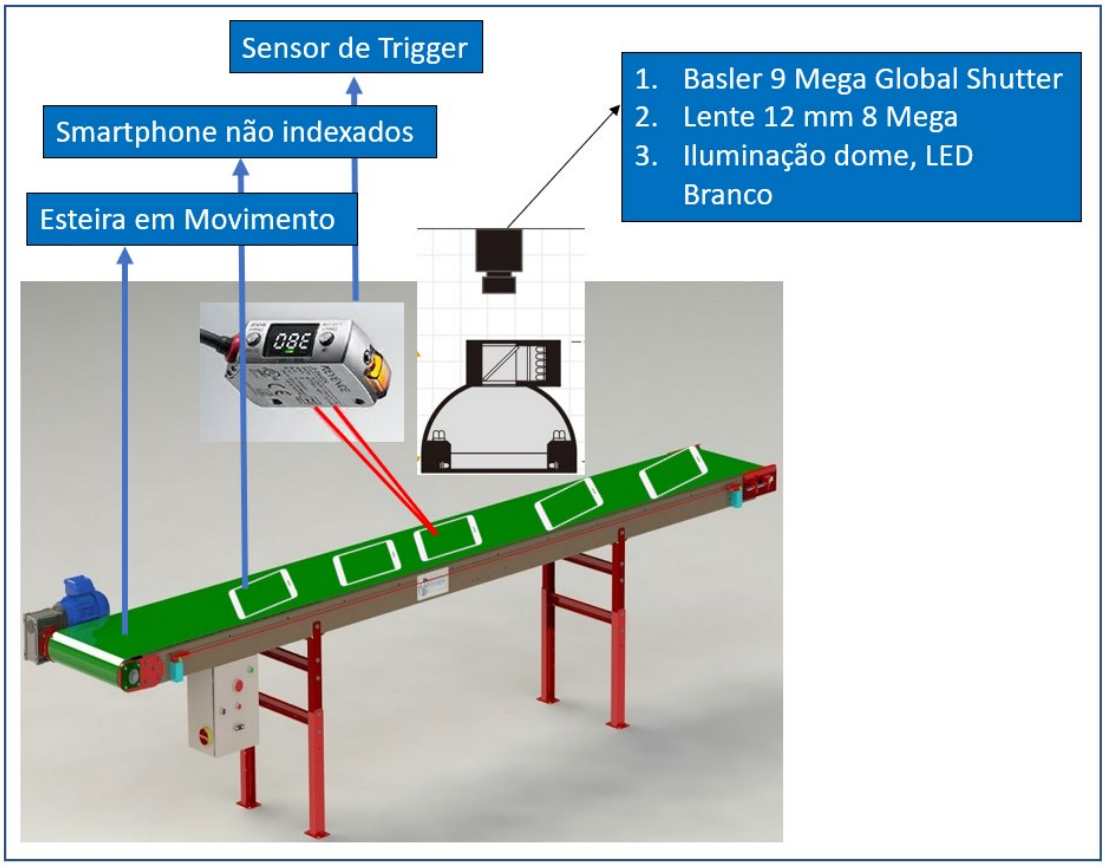


Figura 23 - Sistema de Aquisição de Imagens

Tabela 7 - Características do sistema de visão

Dados de configuração do sistema de visão	
Câmera	Basler acA4096-11gc
Resolução (H x V)	4096 x 2160
Distância de Trabalho (mm)	165
Campo de Visão (mm)	160 x 90
Menor Defeito Horizontal	0,07 mm/pixel
Menor Defeito Vertical	0,08 mm/pixel

O *bracket* possui um *Data Matrix* de 2mm que contém informações, que devem ser inseridas no software de chão de fábrica, de modo a controlar corretamente a produção. Esse *Data Matrix* possui 16 x 16 células, estabelecendo a menor *feature,* a ser identificada, com dimensões de 0,125 mm (2mm/16) na horizontal e vertical.

Para calcular a resolução deve-se considerar, no mínimo 2 pixels para a menor *feature* a ser detectada [78]. A Equação 22 (Fonte [78]) mostra os demais parâmetros necessários para o cálculo da resolução.

$$Resolução = 2\left(\frac{Campo\ de\ visão\ (mm)}{Tamanho\ da\ menor\ feature(mm)}\right) \quad (22)$$

Utilizando o campo de visão especificado na Tabela 7 (90 mm), chega-se a uma resolução de 1440 pixels. O *bracket* está em movimento sobre a esteira, então é necessária uma câmera *Global Shutter*. Além disso, considerou-se uma câmera colorida, pois em alguns modelos de *bracket* é necessário avaliar a cor de alguns componentes.

Os sensores das câmeras coloridas usam um filtro conhecido como Matriz de Bayer, que ocasiona uma perda de resolução na imagem colorida quando comparado com as monocromáticas [79]. Desse modo, resolveu-se utilizar uma câmera com uma resolução maior. O conjunto câmera e lente, especificados na Figura 23, apresentou a melhor relação custo-benefício.

4.2 Materiais

O conjunto de imagens utilizadas para calcular os resultados são referentes a dois modelos de *bracket*, a Tabela 8 apresenta as quantidades de imagens disponíveis.

Tabela 8 - Conjunto de imagens utilizadas na avalição.

Modelo	Imagens sem Defeito	Imagens com Defeito
Modelo A	17	18
Modelo B	54	-

A Figura 24 apresenta o Modelo A e a posição dos 22 componentes avaliados na inspeção visual. Já a Tabela 9 detalha os possíveis defeitos que podem ser encontrados nos componentes, além de apresentar exemplos de anomalias nos componentes.

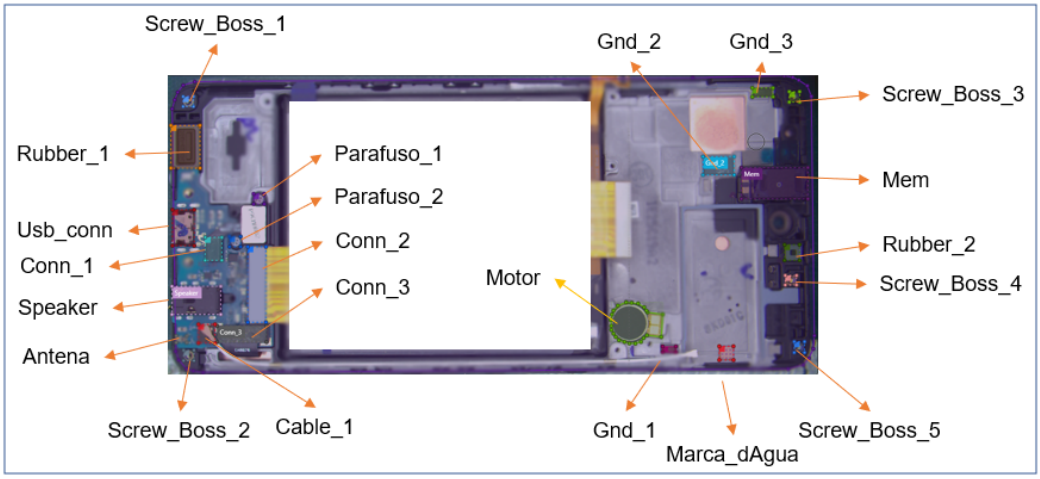
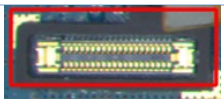
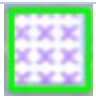
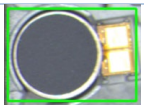
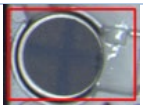
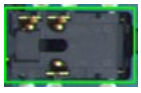


Figura 24 - Modelo A, componentes a serem inspecionados

Tabela 9 – Descrição e exemplo de componentes defeituosos do Modelo A.

Componente	Exemplo Sem Defeito	Descrição Defeitos	Exemplo Defeituoso
Cable_1		Conector ausente, conector não encaixado.	
Conn_1		Conector ausente, conector não encaixado.	
Conn_2		Conector ausente, conector não encaixado.	
Conn_3		Conector ausente, conector não encaixado.	
Gnd_ (1,2,3)		Proteção ausente ou parcialmente removida.	
Marca_dAgua		Violada pela presença de umidade.	
Mem		Componente ausente ou danificado.	
Motor		Componente ausente ou contado danificado.	
Parafuso_ (1,2)		Componente ausente.	
***Rubber*_1**		Componente ausente ou danificado.	
***Rubber*_2**		Componente ausente.	
***ScrewBoss*_ (1,2,3,4,5)**		Componente ausente.	Sem Imagens de Defeitos.
Speaker		Componente ausente ou danificado.	
Usb_Conn		Componente ausente ou danificado.	

A Figura 25 apresenta o *bracket* do Modelo B, também são exibidas as localizações dos componentes que serão segmentados. Para esse modelo será calculado o *IoU Score* para o registro do *bracket* e para a segmentação dos componentes: Motor, Borracha, Parafuso e Conector de Áudio.

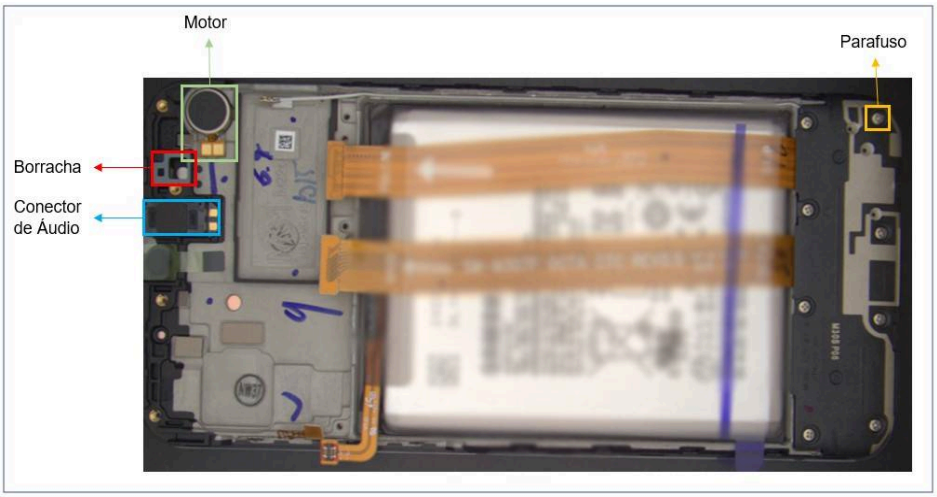


Figura 25 - Modelo B, componentes segmentados.

Além dos dois modelos citados, um terceiro modelo (Modelo C) foi utilizado para apresentar, de forma didática, a sequência de métodos que compõem o comitê. O modelo C não possui nenhuma imagem com defeitos, sendo que os defeitos utilizados nos experimentos foram simulados através de softwares de edição de imagem, como por exemplo o *Paint*. Para o Modelo C estão disponíveis 47 imagens, a Figura 26 apresenta uma imagem do modelo, além de 3 componentes utilizados na análise a ser apresentada neste capítulo.

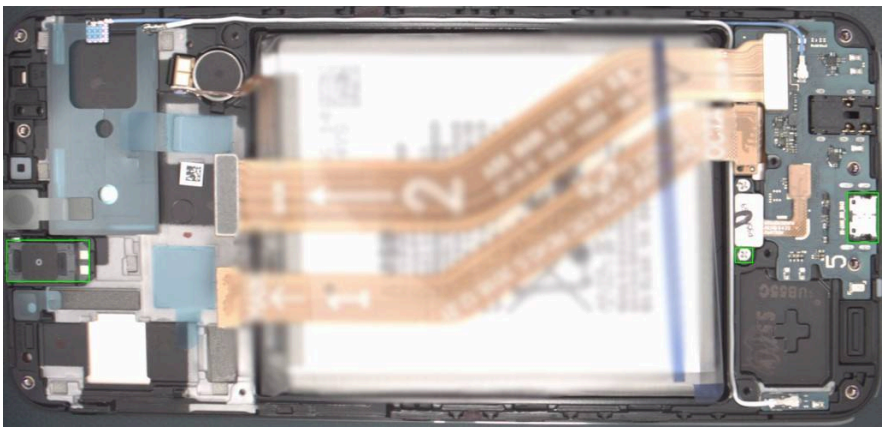


Figura 26 - Modelo C, com 3 componentes marcados.

4.3 Registro do *bracket*

Após a captura das imagens e considerando a possibilidade de rotação e translação das peças, o primeiro passo é realizar o registro, pois uma das características a ser avaliada é o posicionamento dos componentes em relação ao *bracket*.

Existem diversos métodos para proceder o registro da imagem, eliminando assim a rotação e translação em relação à esteira. Inicialmente o fluxo da Figura 27 foi adotado neste trabalho.

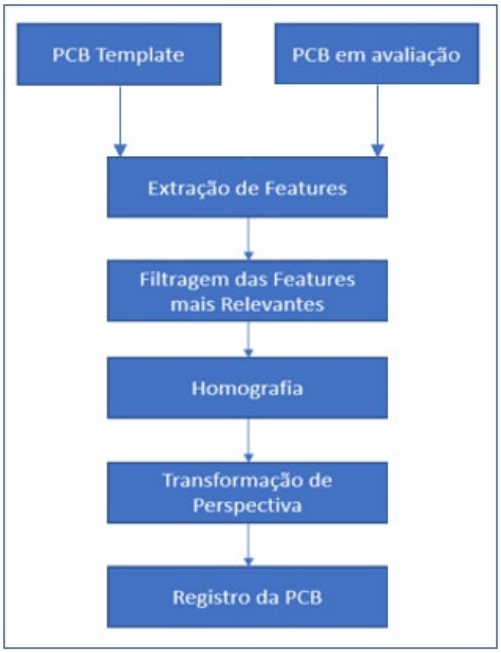


Figura 27 - Fluxo de Registro PCB

Diversos algoritmos, não considerando as abordagens com redes neurais, são propostos para extração de *features*, entre eles: FAST [10], ORB (*Oriented FAST and Rotated BRIEF*) [80], SURF [4] e SIFT[4]. O algoritmo SIFT foi escolhido para desempenhar essa função. Conforme descrito em [4], esse procedimento possibilita que o sistema trate diferentes tipos de PCB, sendo apenas necessário criar um banco de dados de *features* para cada tipo. Além disso, a implementação do algoritmo encontra-se disponível na biblioteca de processamento de imagem escolhida no trabalho, a *opencv* (4.4.0).

Uma vez que as *features* forem detectadas, deve-se encontrar as semelhanças entre a imagem de referência e a imagem da PCB em análise. A partir disso,

transformações na imagem em análise são efetuadas proporcionando o registro da mesma.

O algoritmo SSD foi utilizado em [10] e [81] como forma de estabelecer as correspondências de *features*. Porém, aqui é utilizado o método RANSAC juntamente com a técnica de homografia para efetuar o registro da imagem em análise, conforme proposto em [60].

A biblioteca *opencv* implementa a função *findHomography* [82], que irá calcular a matriz de transformação com uso do algoritmo de RANSAC através dos *keypoints* encontrados pelo SIFT. Tal matriz será utilizada pela função *perspectiveTransform* [83], de modo a criar a máscara utilizada na métrica de avaliação (*IoU Score*) do modelo.

A Figura 28 mostra o resultado da correspondência de *features* retornadas pelo SIFT. À esquerda tem-se a imagem do *bracket* em análise, a ser registrada, enquanto à direita está a imagem de referência. O código que retornou essa imagem foi desenvolvido em Python versão 3.7, a correspondência de *features* foi estabelecida por meio do cálculo da distância Euclidiana entre os descritores retornados, a classe *opencv* responsável pela implementação foi *BFMatcher (Brute Force Matcher)*[84].

Na Figura 29(a) tem-se a detecção da PCB após a aplicação da função *perspectivaTransform* [85], enquanto na Figura 29(b) tem-se a máscara que será utilizada no cálculo do valor *IoU Score* necessário na avaliação do modelo. Por fim, a Figura 30 apresenta as máscaras geradas pela união (30 b) e intersecção (30 a) do *bracket* anotado manualmente pela ferramenta *Vott* e o *bracket* calculado pelo algoritmo de predição proposto. No caso das figuras apresentadas o valor do *IoU* score foi de 0,9600.

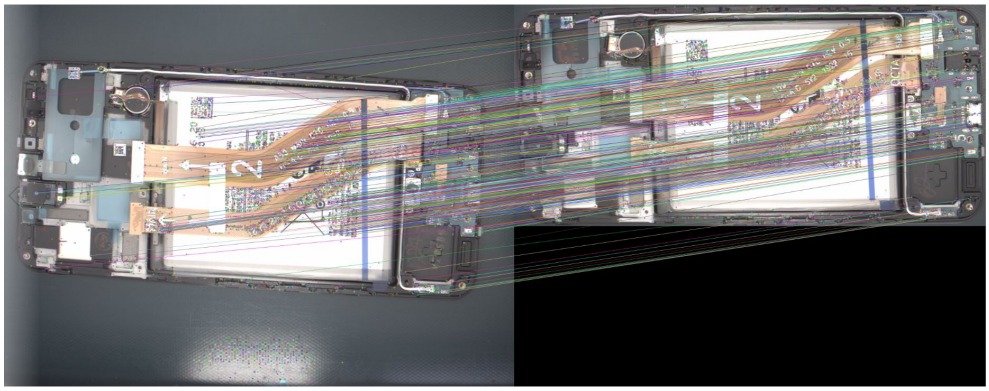


Figura 28 - Correspondência de features

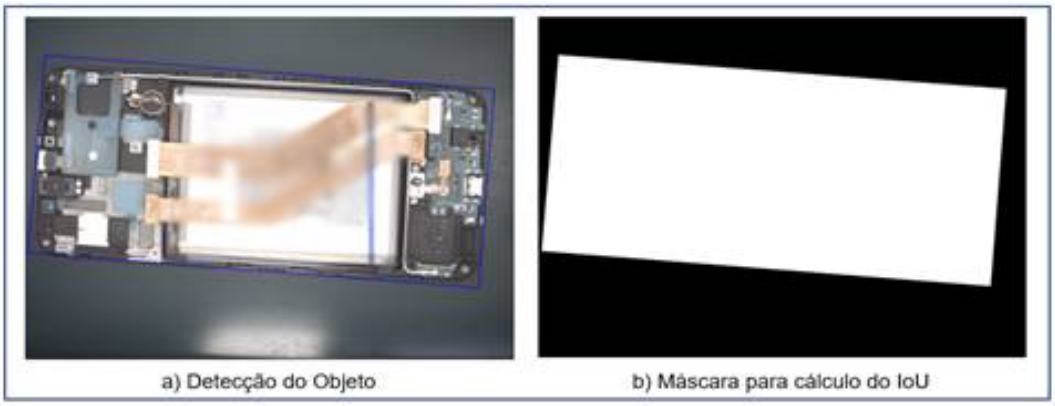


Figura 29 - Resultado da detecção do bracket

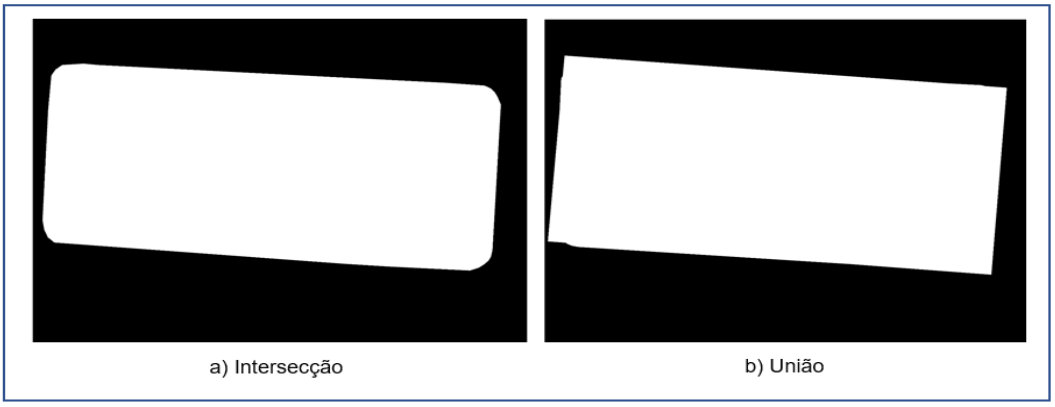


Figura 30 - Cálculo de IoU Score

Uma das dificuldades da abordagem apresentada anteriormente é a seleção das *features* mais relevantes ao problema, pois essa seleção necessita ser bem ajustada manualmente pelos profissionais de processamento de imagem. No caso dos resultados apresentados, após diversos experimentos, considerou-se 80% das *features* resultantes do método *BFMatcher* para o cálculo da matriz de homografia.

Dada a quantidade de modelos de produtos e a velocidade com que novos produtos são lançados no mercado, é necessária uma abordagem mais automatizada na construção do sistema de visão. Atualmente, o DL (*Deep Learning*) está sendo utilizado para resolver problemas complexos, cujas soluções até então eram consideradas inviáveis, apresentando uma alta acurácia e um desempenho

surpreendente quando comparado aos métodos tradicionais de processamento de imagens. A segmentação e detecção de objetos estão entre esses tipos de problemas, onde o DL está sendo aplicado com sucesso [13]. Desse modo, com o uso do DL, ao lançar um novo modelo ou produto, basta apresentar um conjunto de imagens anotadas para o treinamento e o modelo resultante será capaz de segmentar a PCB do produto sobre a esteira.

Porém, os métodos baseados em DL necessitam de um conjunto grande de imagens para treinamento. Caso contrário, o modelo pode não generalizar o suficiente, apresentando baixa acurácia com os dados onde será realizada a predição [13].

No caso de uma linha de produção de produtos eletrônicos, o sistema de inspeção visual não irá dispor de uma quantidade razoável de imagens para o treino, pois o mesmo precisa apresentar alta acurácia desde o início da manufatura do produto. Para solucionar esse problema uma técnica comumente utilizada é o DA (*Data Augmentation*) [6], [11], [13], [48].

A rede neural proposta em [48] será usada para gerar um modelo capaz de detectar a PCB usando a técnica de segmentação semântica. Os resultados da U-Net serão comparados com aqueles encontrados pelo método descrito no fluxo apresentado na Figura 27, ou seja, o registro do *bracket* utilizando-se a técnica de homografia. O método que apresentar o melhor *IoU Score* será utilizado no restante do trabalho.

Considerando a técnica de segmentação com a U-Net, o fluxo da Figura 27 será alterado, sendo necessária a adoção de uma fase de treinamento. Após realizado o treinamento, o registro da imagem é feito utilizando-se da máscara detectada pela U-Net e da transformação afim que irá remover qualquer rotação. A Figura 31 traz o fluxo descrito e as funções *opencv* utilizadas.

Todo o código fonte desenvolvido neste trabalhou utilizou a linguagem *Python*. O computador utilizado nos testes possui sistema operacional Windows 10, 8 gigabytes de memória RAM (*Random Access Memory*) e placa de vídeo NVIDIA GeForce GTX 1660 Ti. O treinamento utilizando a GPU (*Graphics Processing Unit*) citada foi feito utilizando-se do Tensorflow versão 2.2.0.

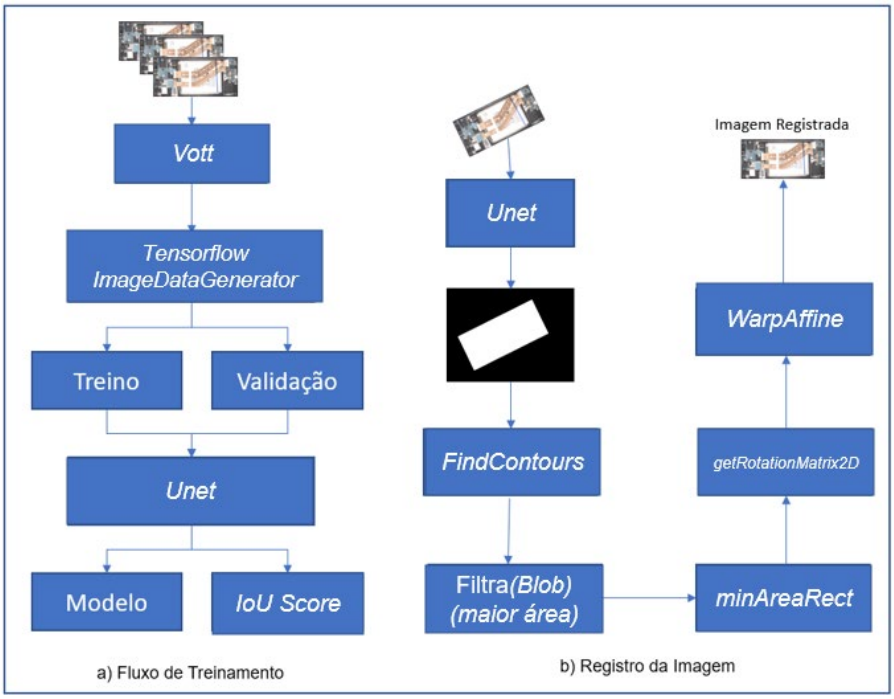


Figura 31 - a) Fluxo de Treinamento, b) Registro da PCB com auxílio das funções opencv.

O conjunto de imagens, utilizando o Modelo C de *bracket*, foi dividido em 2 outros conjuntos: treino com 32 imagens e teste com 15 imagens. Ambos os conjuntos foram aumentados, criando-se novas imagens sintéticas através classe *ImageDataGenerator* [86] do pacote *tf.keras.preprocessing.image*. Os parâmetros utilizados na geração das imagens refletem as condições encontradas no ambiente de teste, que são baseadas na linha de produção real. Tais parâmetros são apresentados na Tabela 10.

Tabela 10 - Parâmetros de configuração da classe ImageDataGenerator

Parâmetros do DA		
Parâmetro	**Descrição**	**Valor**
rotation_range	Limite de graus que são usados para selecionar um valor randômico.	5
width_shift_range	Fração da largura total que será selecionada, de forma randômica, na translação.	0,02
height_shift_range	Fração da altura total que será selecionada, de forma randômica, na translação.	0,02

Após configurada a classe que faz a geração das imagens sintéticas, é necessário configurar quais imagens serão consideradas nesse aumento de dados. O método *flow_from_directory* da classse *ImageDataGenerator* foi utilizado para esse fim, de modo a criar uma base de teste de 150 imagens, além de ser utilizado para alimentar o treino da U-Net gerando 32 imagens para cada época do treinamento.

A Tabela 11 apresenta as configurações usadas no treinamento da U-Net, enquanto a Tabela 12 traz o *IOU Score*, junto com o tempo médio de processamento, dos dois métodos utilizados.

Tabela 11 - Dados de Treinamento e Data Augmentation

Dados do DA				
Imagens Originais	**Tamanho do Batch**	**Rotação (Graus)**	**Translação Vertical (%)**	**Translação Horizontal (%)**
32	32	5	0.02	0.02
Dados do treino da U-Net				
Épocas	***Learn Rate***	**Otimizador**	***Loss***	**Métrica**
50	0,01	Adam	*Binary Crossentropy*	*Accuracy*

Tabela 12 - Comparação entre Homografia e U-Net

	Técnica	***IoU Score***	**Tempo (s)**
Modelo A	U-Net	0,9787 ±0,0154	0,1274 ±0,4447
	Homografia	0,9545 ±0,0126	2,5123 ±0,1164
Resultados obtidos com 150 imagens de teste.			

Conforme mostrado na Tabela 12, a U-Net apresentou um melhor desempenho em relação à homografia. Deste modo, U-NET foi escolhida para ser utilizado no registro. A Figura 32(a) exibe o resultado da detecção do *bracket* sobre a esteira, já a Figura 32(b) mostra o resultado final do registro do *bracket*.

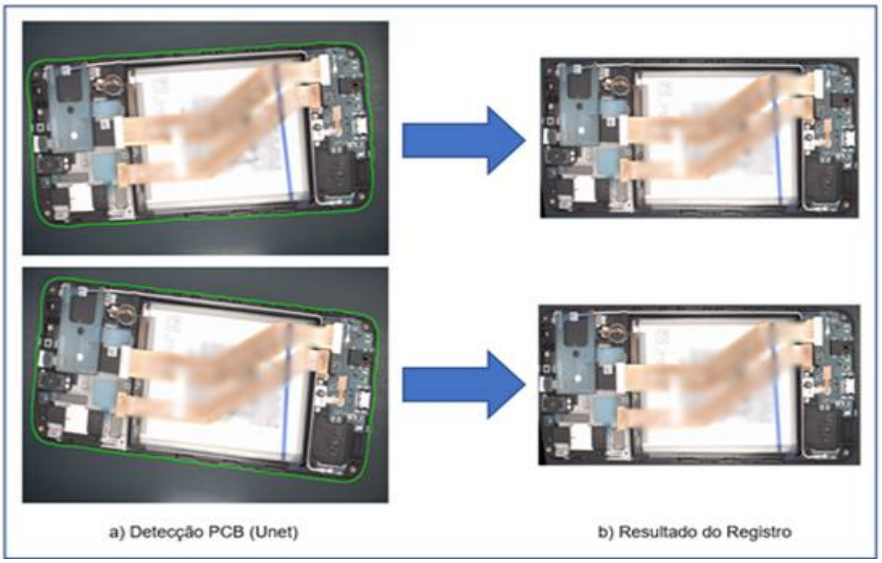


Figura 32 - Resultado do Registro, após detecção com a U-Net

Mesmo com o registro apresentando alto *IoU Score*, deve-se tentar melhorá-lo, pois qualquer variação de posicionamento pode resultar em perda de acurácia na detecção dos componentes. Desse modo, considerando que a segmentação semântica apresenta baixa acurácia nas bordas de uma PCB conforme apresentado em [68], foi acrescentado o algoritmo *watershed* à U-Net. A Figura 33 mostra a sequência de operações referentes à junção dos dois métodos, mostrando também o uso de marcadores para controlar a segmentação pelo algoritmo *watershed*.

A biblioteca opencv implementa o algoritmo *watershed* controlado por marcadores através da função de mesmo nome do algoritmo. Os marcadores necessários para o seu uso devem ser parametrizados da seguinte maneira:

- Os pixels da região que representa o fundo (*background*), esteira no caso do trabalho, devem receber um determinado valor diferente de 0.
- Os pixels da região que representa o *bracket* (*foreground*) devem receber um valor diferente de 0 e diferente do valor atribuído ao fundo.
- Os pixels referentes a região desconhecida devem possuir valores igual a 0.

A função *watershed* recebe como parâmetros os marcadores e a imagem original e consegue estabelecer as fronteiras entre os objetos (*background* e *foreground)* melhorando assim a segmentação iniciada pela U-Net.

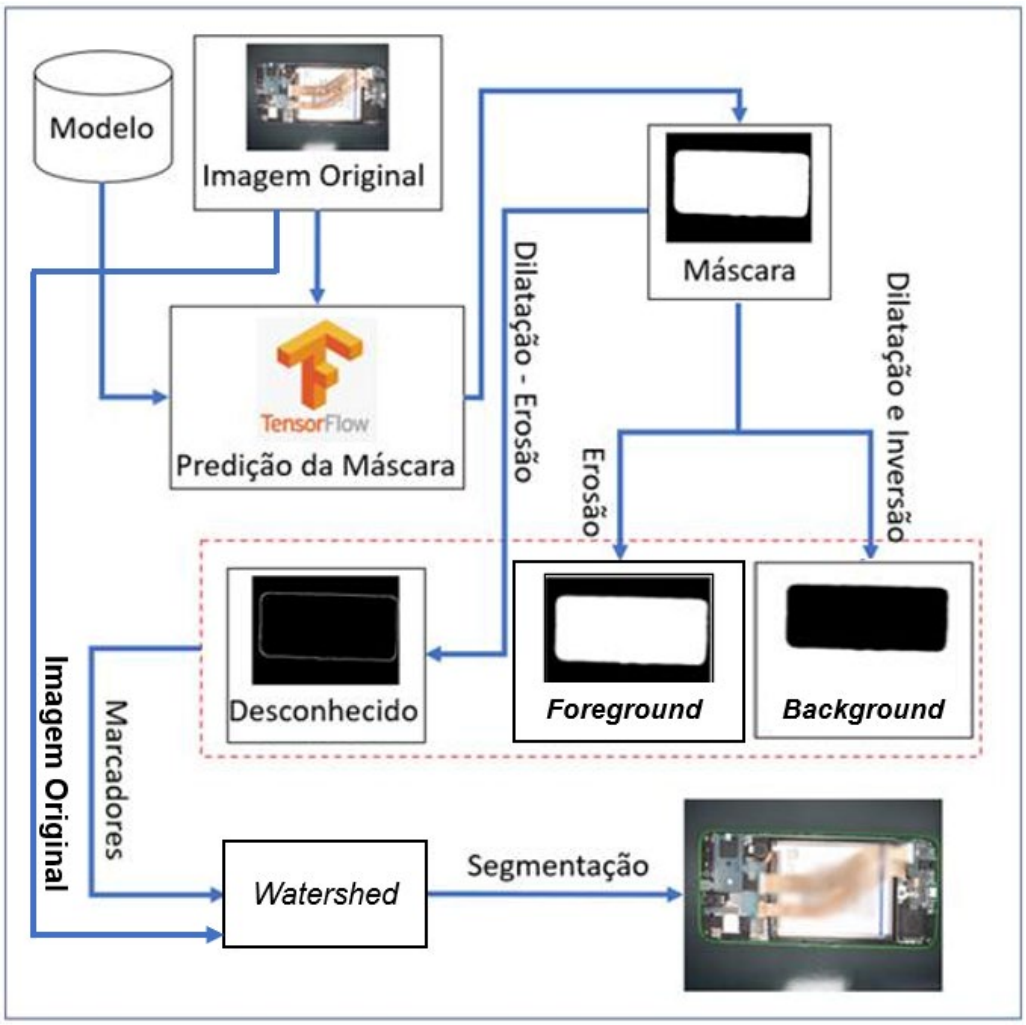


Figura 33 - Junção Watershed e U-Net

Diversos elementos estruturantes foram testados nas operações de erosão e dilatação utilizado para estabelecer os marcadores, sendo utilizado um software em *Python* de forma a encontrar a melhor combinação. Após diversos testes, o elemento estruturante quadrado 10 x 10 apresentou os melhores resultados. Considerando esse elemento, o algoritmo *watershed* elevou o *IoU Score* de 0,9787 para 0,9856. A diferença entre o *IoU Score* encontrado com o uso do *watershed* e o *IoU Score* encontrado a partir da U-Net não é expressiva, porém nas próximas seções será discutido a segmentação de componentes, onde o uso do *watershed* melhora de forma expressiva o resultado da segmentação.

4.4 Detecção de anomalias

4.4.1 Segmentação dos componentes

Uma vez que a PCB do produto está devidamente registrada, segue-se com a análise de anomalias dos componentes. Considerando o *IoU Score* de 0,9856 para o registro do *bracket* apresentado na seção anterior, inicialmente será verificado o quanto essa pequena variação no registro interfere na localização dos componentes.

Para esse primeiro teste foram selecionados 3 componentes do Modelo C de bracket. As coordenadas dos retângulos que delimitam tais componentes, além do retângulo delimitador do próprio *bracket*, foram estabelecidas manualmente utilizando-se o software Vott (etapa 1). Tais coordenadas foram formatadas segundo o padrão JSON (*JavaScript Object Notation*) e exportadas (etapa 2).

O arquivo exportado pelo Vott foi analisado por um programa em *Python (*etapa 3). Esse programa criou a máscara para cada um dos componentes, além de proporcionar o registro do *bracket* eliminando a rotação e translação das imagens (etapa 4).

A Figura 34 exemplifica o processo descrito, organizando o procedimento em etapas numeradas de 1 a 4. Essa sequência de 3 máscaras foi gerada para cada uma das imagens em teste.

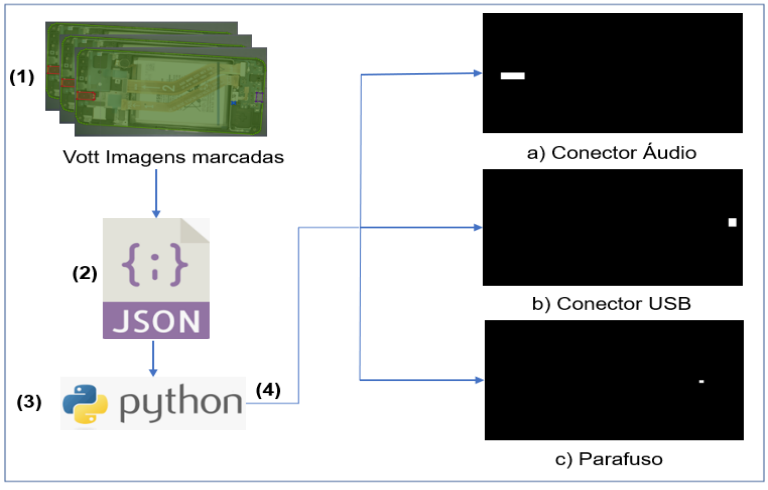


Figura 34 - Exemplo das máscaras geradas

Para criar as coordenadas de referência para cada um dos componentes, novamente o software Vott foi utilizado. Uma imagem de referência sem rotação e

translação foi marcada cuidadosamente, sendo possível então extrair as coordenadas dos retângulos delimitadores de cada componente. Essas coordenadas serão utilizadas para criar as máscaras no passo seguinte.

Aproveitando-se do modelo criado para o registro utilizando a junção *watershed* e U-Net, foi feito o registro de cada um dos *brackets*. Então, utilizando-se das coordenadas de referência, criou-se as máscaras. A Figura 35 resume o processo descrito.

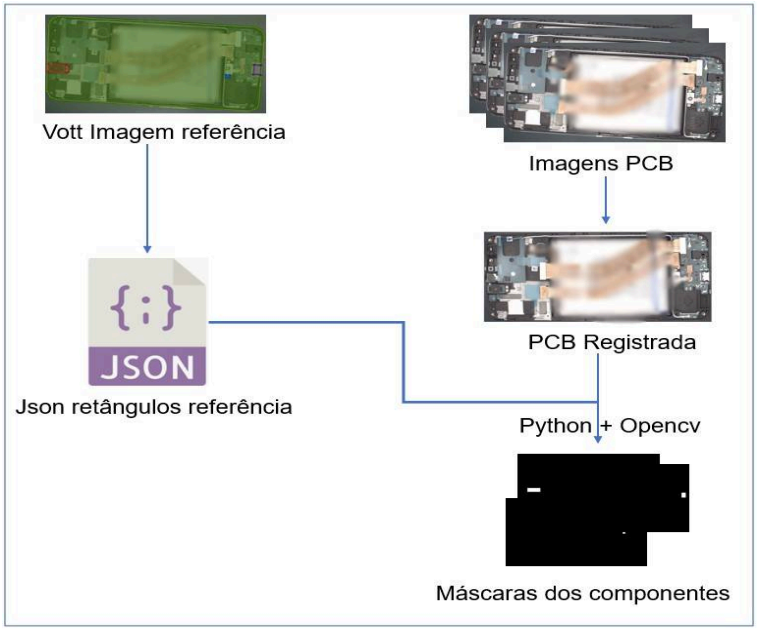


Figura 35 - Processos de criação de máscaras

Um outro programa *Python* calculou o *IoU Score* a partir dos conjuntos de máscaras criados. A Tabela 13 apresenta os resultados encontrados.

Tabela 13 - IoU Score Médio após registro das PCB

Total Imagens	**5**
Componente	**IoU Score Médio**
Conector Áudio	0,8390
Parafuso	0,5580
Conector Usb	0,8200

Conforme mostrado na Tabela 13, o componente Parafuso não teve a sua posição determinada com alta acurácia (*IoU Score* de 0,5580). Deste modo,

operações de subtração da imagem de teste com uma imagem referência não iria trazer bons resultados. A Figura 36 mostra um exemplo, onde dois componentes: conector áudio e parafuso não tiveram seus retângulos delimitadores localizados corretamente.

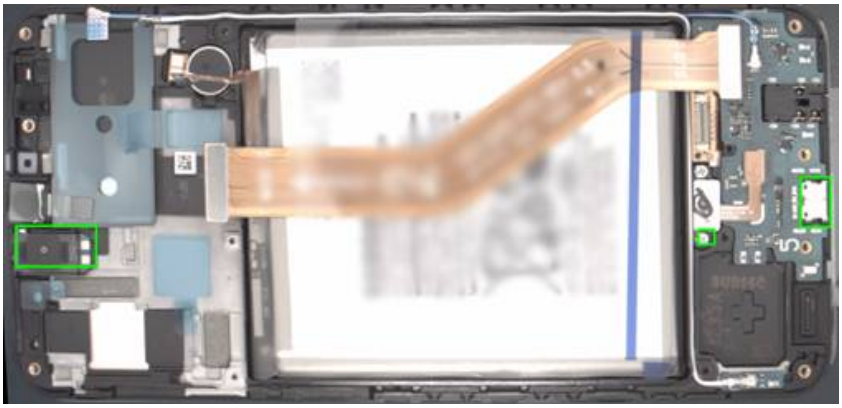


Figura 36 - Erro na determinação da localização dos componentes (Modelo C).

Visando melhorar a acurácia na segmentação do componente, uma nova U-Net será utilizada para segmentar cada componente dentro de uma região específica. A partir dos retângulos delimitadores dos componentes, as regiões serão recortadas da imagem original (será considerado um incremento configurável na área de pesquisa) e as máscaras dos componentes definidas. Essas imagens servirão de base para o aumento da base de dados, a serem utilizadas no treinamento de uma U-Net para cada componente. A Figura 37 exibe as regiões recortadas e as máscaras resultantes para cada um dos componentes.

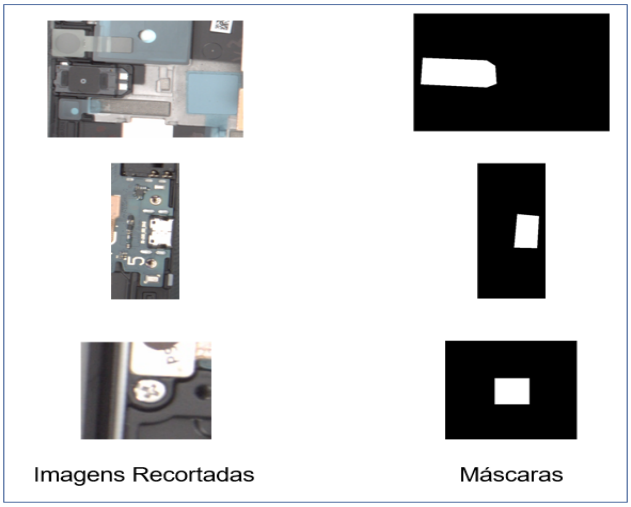


Figura 37 - Imagens Recortadas e Máscaras Geradas

No processo de treinamento, utilizou-se a DA por meio do mesmo código em *Python* e configurações utilizadas no treinamento da U-Net para segmentar o *bracket*, que foram apresentados na Tabela 11 deste trabalho.

O modelo U-Net para cada componente foi utilizado para substituir as coordenadas de referência, modificando o fluxo apresentado na Figura 35. Novamente, as máscaras dos componentes encontradas pelo novo processo, foram submetidas ao cálculo do *IoU Score*. A Tabela 14 exibe os *IoU Scores* médios alcançados com o emprego da técnica descrita.

Tabela 14 - IoU Score após U-Net dos componentes

Total Imagens	**5**
Componente	**IoU Score Médio**
Conector Áudio	0,876
Parafuso	0,876
Conector Usb	0,781

A Figura 38 mostra os retângulos delimitadores encontrados a partir do registro do *bracket* e da aplicação da junção U-Net e *watershed* para cada um dos componentes. É visível que houve uma melhora na detecção das coordenadas dos componentes quando se compara a Figura 38 com a Figura 36.

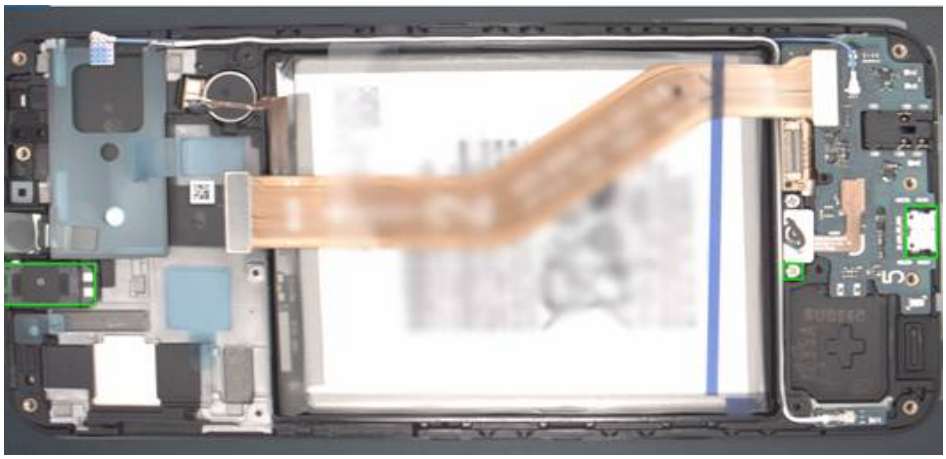


Figura 38 - Retângulos dos componentes encontrados com a U-Ne (Modelo C).

4.4.2 Detecção de anomalias (Métodos Tradicionais)

Nesta subseção, são apresentados dois métodos que não fazem uso da IA (Inteligência Artificial) para avaliar a presença de anomalias nos componentes do *bracket*.

4.4.2.1 Subtração em relação à imagem de referência (template).

Segmentado o componente, com o uso de duas redes U-Net, a subtração da imagem em teste em relação à uma imagem *template* será utilizada para detectar anomalias em cada um dos componentes. Se as anomalias não existirem, espera-se que a diferença entre as imagens tenha área igual a zero.

A Tabela 15 apresenta a diferença média encontrada considerando os 3 componentes iniciais do estudo. Enquanto que a Figura 39 mostra um exemplo das diferenças encontradas.

Tabela 15 - Área das diferenças

Total Imagens	**5**
Componente	**Área média da diferença (%)**
Conector Áudio	1,162
Parafuso	4,656
Conector Usb	1,992

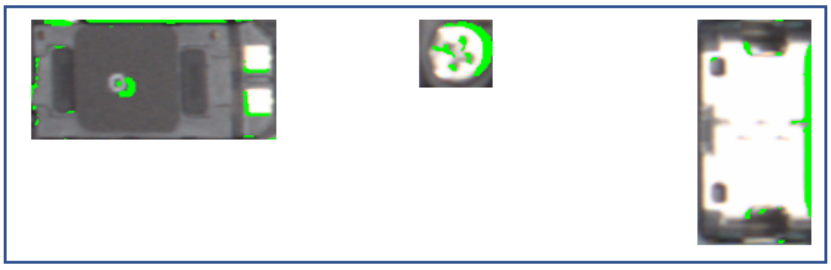


Figura 39 - Exemplo de diferença para cada um dos componentes

Uma das dificuldades desse método é estabelecer um limiar de área, após a subtração, que identifique um componente com anomalia. Neste trabalho, será utilizado o método do desvio padrão, onde um *outlier* (anomalia) é definido pela área que estiver *n* desvios padrões de distância da média. O parâmetro *n* será estabelecido através de exemplos de imagens sem defeitos.

Essa análise estatística, para estabelecer o número de desvios padrões necessários para estabelecer o limiar de aprovação, também representa uma dificuldade para o uso desse método. Além disso, algumas variações, como por exemplo a cor dos componentes são aceitas, sendo inclusive necessário atualizar o *template* de acordo com o lote em produção.

4.4.2.2 Reconstrução baseada na 2D DFT (2D Discrete Fourier Transform)

Considerando as variações dos produtos aceitos nas indústrias eletrônicas, este trabalho também irá considerar um outro método baseado na 2D DFT (2D *Discrete Fourier Transform*). Segundo o que é apresentado em [21], esse método consegue detectar e localizar pequenos defeitos em PCBs, sendo invariante à iluminação e às variações aleatórias dos produtos.

Este método também é baseado na comparação com uma imagem padrão (*template*), sendo necessário, assim como o método anterior, atualizar a imagem padrão de acordo com o lote de produção.

A Figura 40 exibe os 2 passos, utilizados para encontrar anomalias através da reconstrução baseada na 2D DFT:

1. A imagem em teste e a imagem padrão do item em inspeção são submetidas à 2D DFT, resultando nas matrizes F_T e F_I que são as suas respectivas transformadas de Fourier. Os espectros de potências dessas duas matrizes também são calculados, após a conversão para coordenadas polares tem-se as matrizes P_T e P_I.
2. Para cada ângulo e raio das matrizes em coordenadas polares, é calculada um nova matriz F' aplicando-se as regras da Figura 40 (regras a,b e c). A nova matriz, que é construída utilizando as matrizes F_T e F_I, representa a 2D DFT da imagem reconstruída. Nesse processo de reconstrução, as componentes semelhantes são descartadas, espera-se que somente as componentes de anomalias sejam mantidas.

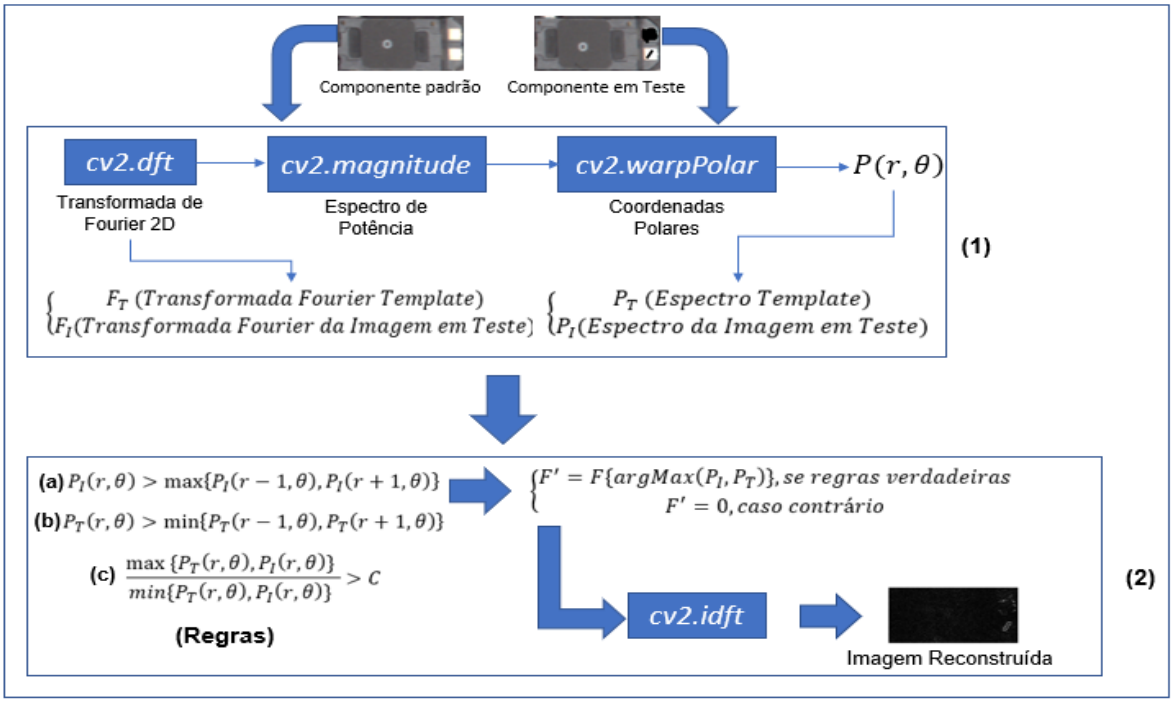


Figura 40 - Aplicação da reconstrução da imagem baseada na transformada de Fourier 2D.

Com a nova matriz calculada, aplica-se a DFT inversa 2D, reconstruindo a imagem. Finalmente, é aplicado um limiar adaptativo, encontrando-se os *blobs*, regiões que representam a diferença da imagem em teste em relação a imagem padrão.

A Figura 41 apresenta um exemplo de detecção de anomalia utilizando a reconstrução a partir da DFT 2D. Esse método apresenta algumas dificuldades: selecionar um limiar de binarização capaz de diferenciar regiões defeituosas do ruído, relativo à reconstrução, e determinar o valor de *C* (regras da Figura 40). O valor *C* representa a relação entre o maior e o menor espectro de potência das imagens em análise.

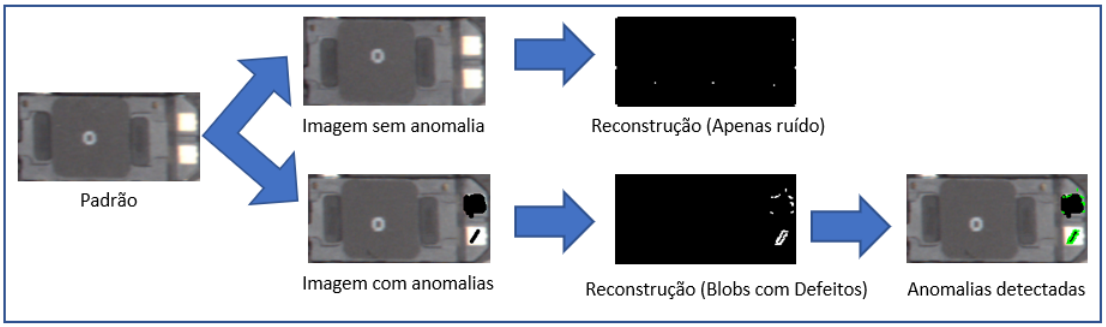


Figura 41 - Exemplo de reconstrução a partir da transformada de Fourier.

4.4.3 Detecção de Anomalias (Métodos que fazem uso de IA)

Além dos métodos tradicionais já descritos, o presente trabalho também fará o uso de métodos baseados em DL. Um dos problemas do DL é o número de imagens para o treinamento, conforme já citado, pois na indústria eletrônica não são normalmente geradas muitas imagens com defeitos. Assim, o trabalho irá utilizar métodos baseados no aprendizado não supervisionado, conforme será apresentado nas próximas subseções.

4.4.3.1 Agrupamento baseado nas features extraídas pela VGG

O método criado em [63] é baseado no TL (*Transfer Learning)* da rede VGG16 pré-treinada com o conjunto de imagens ImageNet [87]. As 2 camadas totalmente conectadas e a camada *softmax* da rede original são removidas, sendo substituídas por duas camadas totalmente conectadas com 1000 neurônios cada e uma camada *softmax* com o número de neurônios igual ao número de classes. Neste caso igual ao número de componentes que irão participar do aprendizado supervisionado. As camadas convolucionais originais da VGG16 não serão retreinadas.

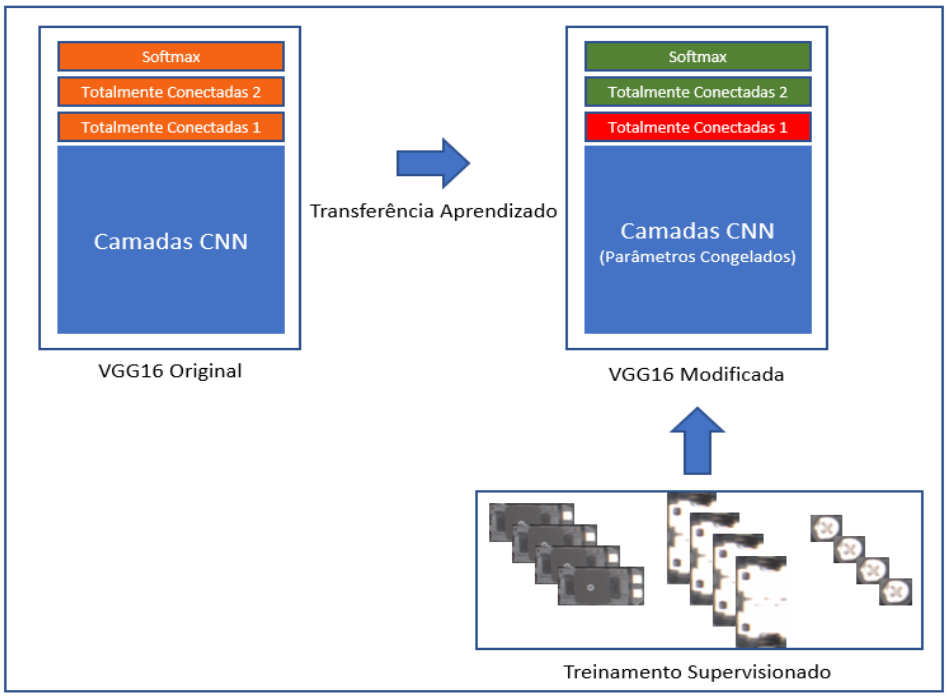


Figura 42 – Transfer learning e fine-tunning de Parâmetros.

A Figura 42 mostra os detalhes desse processo, os retângulos em cor laranja representam as camadas que serão removidas da rede original, enquanto que os retângulos em verde e vermelho representam as camadas que serão retreinadas na rede modificada. Como trata-se de um TL, não é necessário um grande número de imagens dos componentes para encontrar os melhores parâmetros para as novas camadas.

Uma vez treinada a rede, estabelece-se um agrupamento utilizando-se das *features* da primeira camada totalmente conectada (representada pelo retângulo vermelho na Figura 42), após as camadas convolucionais. Essas *features* servirão de entrada para o algoritmo de agrupamento *K-means*, sendo de responsabilidade desse algoritmo fornecer os centroides dos grupos formados para cada componente. A Figura 43 exemplifica o processo descrito, o desenho de um *X*, em vermelho, mostra as camadas que serão descartadas da rede VGG16 modificada.

Após estabelecidos os centroides, é possível calcular a distância euclidiana entre as *features* de cada imagem com seu respectivo centroide. Assim, as imagens são consideradas anômalas quando essa distância for maior que um determinado limiar. Novamente, o presente trabalho irá considerar a média somada a *n* desvios padrões, sendo necessário um conjunto mínimo de imagens sem defeito para esse cálculo estatístico.

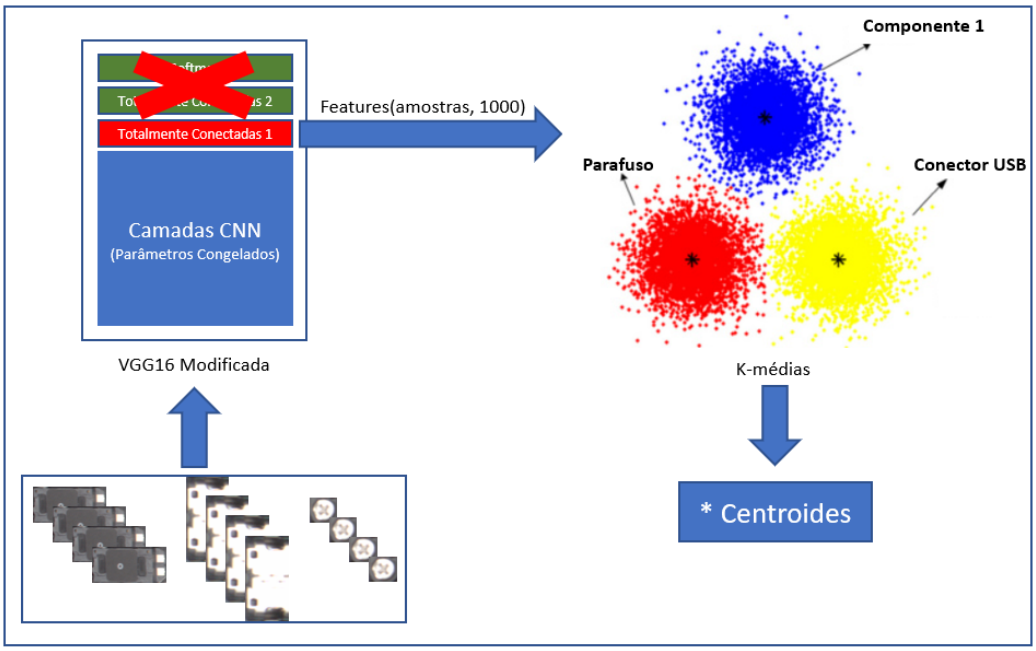


Figura 43 - Cálculo dos Centroides para os componentes

Para exemplificar o uso do método, novamente os 3 componentes Conector de Áudio, Conector de USB e Parafuso serão utilizados. Além disso, foram criados 3 exemplos de defeito (Tabela 16) para exemplificar o cálculo do limiar de aprovação.

Tabela 16 - Teste de detecção com 3 grupos

Quantidade de imagens: 5 imagens de cada componente.		
Coeficiente de silhueta do agrupamento: 0,83		
Componente	**Distância média do centroide**	**Desvio padrão da distância.**
Conector de Áudio	49,998	15,123
Parafuso	45,404	7,729
Conector USB	35,557	9,712
Distância componentes defeituosos		
Componente	**Exemplo**	**Distância do centroide**
Conector de Áudio		105,010
Parafuso		139,179
Conector USB		383,658

Conforme mostrado pela Tabela 16, estabelecendo-se um limiar igual a média da distância acrescida de no mínimo 2 desvios padrões, consegue identificar os componentes com anomalias nos 3 grupos de componentes apresentados.

4.4.3.2 CAE (Convolutional AutoEncoder)

Continuando com a busca por métodos baseadas em aprendizado não supervisionado, o CAE também será considerado. Esse método foi apresentado nos trabalhos descritos em [6], [70], [88], porém somente em [70] o CAE é utilizado na detecção de anomalias em PCB, sendo que o presente trabalho irá se basear na metodologia apresentada por este.

Ambas as partes *encoder* e *decoder do CAE*, serão compostas por 4 camadas. As camadas do *encoder* serão compostas pelas seguintes sub camadas: camada de

convolução 2D, camada de normalização, camada de ativação e camada *Max Pooling*. Já o *decoder* apresenta uma estrutura semelhante de sub camadas, sendo que a camada de *Max Pooling* é substituída por uma cada de *Up Sampling*. A Figura 44 e a Figura 45 mostram o código fonte escrito utilizando a *API Keras*, com detalhamento dos parâmetros de configuração das camadas citadas.

```
input_img = keras.Input(shape=(128,128,1))
filter_size = (3,3)
filters = 8
#Camada 1
x = keras.layers.Conv2D(filters * 8,filter_size, activation='relu', padding='same')(input_img)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.MaxPooling2D((2,2), padding='same')(x)
#Camada 2
x = keras.layers.Conv2D(filters * 16,filter_size, activation='relu', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.MaxPooling2D((2,2), padding='same')(x)
#Camada 3
x = keras.layers.Conv2D(filters * 32,filter_size, activation='relu', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.MaxPooling2D((2,2), padding='same')(x)
#Camada 4
x = keras.layers.Conv2D(filters * 64,filter_size, activation='relu', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
encoded_layer = keras.layers.MaxPooling2D((2,2), padding='same')(x)
```

Figura 44 - CAE encoder, keras API

```
#Camadas decoder
#Camada 1
x = keras.layers.Conv2D(filters * 64, filter_size, activation='relu', padding='same')(encoded_layer)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.UpSampling2D((2, 2))(x)
#Camada 2
x = keras.layers.Conv2D(filters * 32, filter_size, activation='relu', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.UpSampling2D((2, 2))(x)
#Camada 3
x = keras.layers.Conv2D(filters * 16, filter_size, activation='relu', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.UpSampling2D((2, 2))(x)
#Camada 4
x = keras.layers.Conv2D(filters * 8, filter_size, activation='relu', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.UpSampling2D((2, 2))(x)

decoded = keras.layers.Conv2D(1, filter_size, activation='sigmoid', padding='same')(x)
```

Figura 45 - CAE decoder, Keras API

No treinamento do CAE apenas as imagens dos componentes sem anomalias serão utilizadas. Para conseguir as imagens dos componentes, será feito o registro do *bracket* e, consequentemente, a segmentação dos componentes utilizando a junção *U-Net* com *watershed* já apresentada.

Com o treinamento realizado, o CAE será capaz de extrair as principais *features* do componente e reconstruir a imagem do componente por meio destas. Quando um componente com defeito for apresentado ao CAE, a diferença entre a imagem reconstruída e a original será maior que um determinado limiar. Como forma de avaliação dessa diferença será utilizado o MSE *(Mean Squared Error*) e o SSIM *(Structural Similarity Index Measure*). A Figura 46 exemplifica o processo com o CAE.

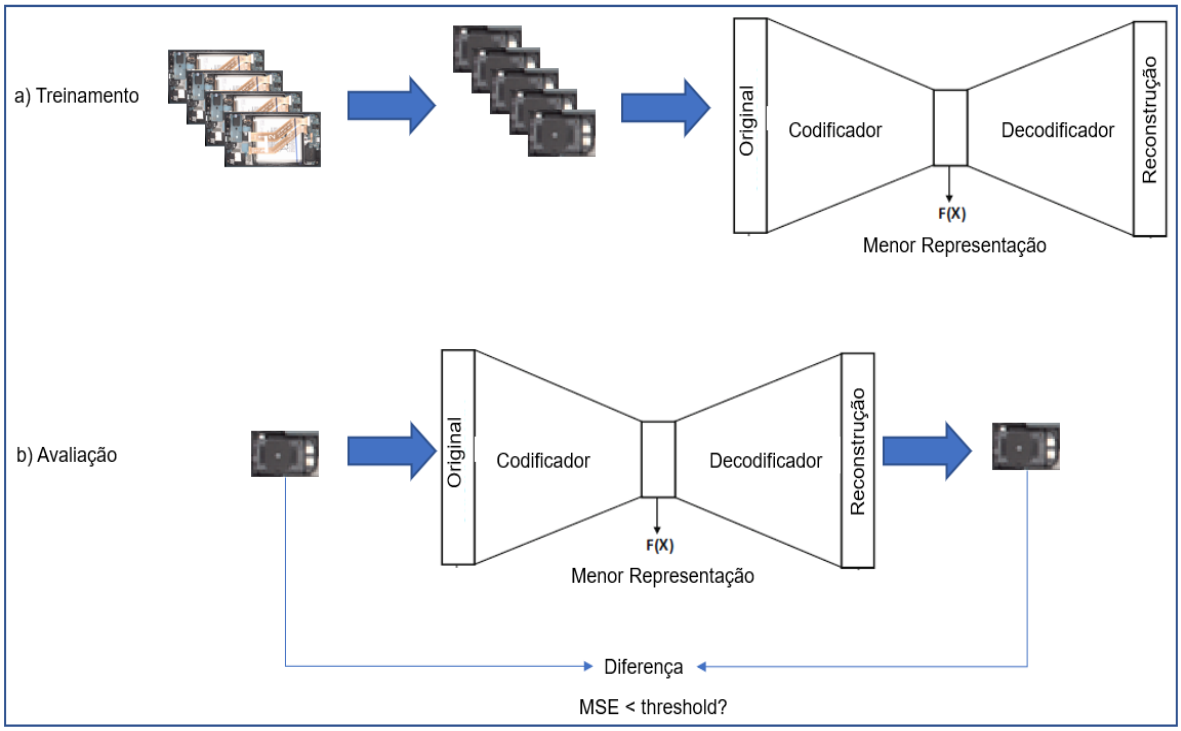


Figura 46 - Avaliação com CAE

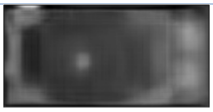
A Tabela 17 apresenta o MSE e o SSIM considerando apenas as imagens sem anomalias. O processo de cálculo está apresentado na Figura 46 (b).

Tabela 17 - MSE, SSIM componentes sem anomalia

Quantidade de imagens: 5 imagens de cada componente.		
Componente	**MSE (Desvio Padrão)**	**SSIM (Desvio Padrão)**
Conector de Áudio	88,7308 (6,2606)	0,8504 (0,0041)
Parafuso	88,3243 (16,0295)	0,9507 (0,0134)
Conector USB	112,7689 (4,3176)	0,9225 (0,0047)

A Tabela 18 exibe os resultados do teste considerando componentes com anomalias. Pelos valores apresentados na tabela é possível estabelecer um limiar de aprovação para todos os exemplos.

Tabela 18 - MSE, SSIM componentes com anomalias

Exemplo componentes com anomalias.				
Componente	**Original**	**Reconstruída**	**MSE**	**SSIM**
Conector de Áudio			141,5656	0,7928
Conector USB			390,5496	0,7702
Parafuso			56,8823	0,8336

4.4.3.3 WGAN-GP (Wasserstein GAN with Gradient Penalty)

Ainda considerando o fato de ser difícil encontrar imagens de componentes defeituosos para o treinamento, o presente trabalho irá considerar uma rede GAN, mais especificamente uma WGAN-GP para encontrar anomalias nos componentes conforme proposto nos trabalhos [11], [65].

Em [65], além da rede WGAN-GP o autor propõe o uso do TL para extrair as *features* utilizadas no treinamento dos componentes da rede: crítico e o gerador. Deste modo, o TL é feito através dos pesos da *Inception V3* treinada com o conjunto de dados *ImageNet*, sendo necessárias poucas imagens de componentes não defeituosos para o treinamento do modelo. A Figura 47 representa o método a ser desenvolvido.

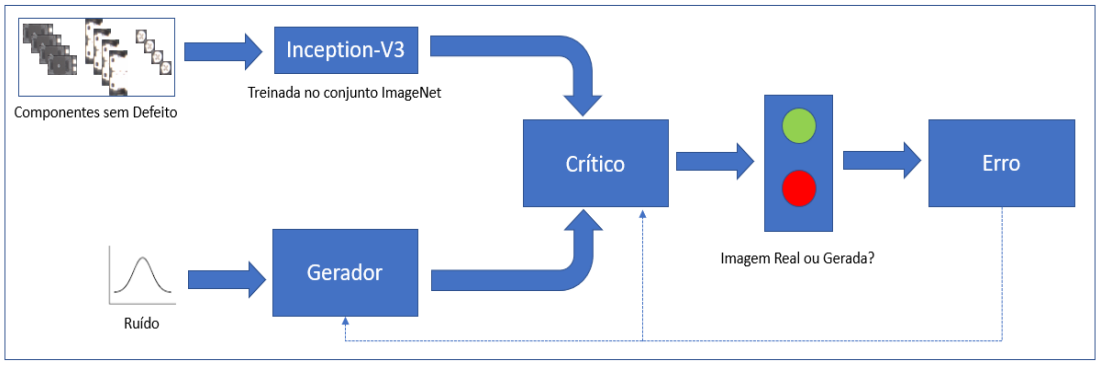


Figura 47 - WGAN-GP com transfer learning

O modelo referente ao crítico utiliza uma função linear como saída, sendo assim, após o treinamento da GAN é necessário verificar qual o valor resultante para cada conjunto de componentes sem defeito. Baseando-se na média e desvio padrão da saída do crítico, estabelece-se um valor limiar de aprovação para cada componente a ser inspecionado.

Nos testes iniciais considerando, os componentes utilizados até o momento, a extração de *features* através da *Inception* não apresentou bons resultados. Deste modo, decidiu-se abandonar o TL para esse método.

Tabela 19 - Saída Crítico para componentes não defeituosos

WGAN-GP: Componentes sem defeito			
Componente	**Imagem**	**Média saída do Crítico**	**Desvio Padrão saída do Crítico**
Conector de Áudio		28,2917	0,2274
Conector USB		5,9616	4,1650
Parafuso		-47,7924	4,8370

Tabela 20 - Saída Crítico para componentes defeituosos

WGAN-GP: Componentes com defeito			
Componente	**Imagem**	**Saída do Crítico**	**Intervalo (2 desvios padrão)**
Conector de Áudio		26,2218	[27,9506; 28,7465]
Conector USB		-26,5241	[-2,3684; 14,2916]
Parafuso		-22,6949	[-57,4664; -38,1184]

Na Tabela 19 tem-se a média e desvio padrão da saída do crítico, após o treinamento da GAN, para os componentes não defeituosos. Com esses valores, é estabelecido o intervalo de aprovação para os componentes (média da saída do crítico ± 2 x desvio padrão), conforme apresentado na Tabela 20. É possível perceber que esse modelo conseguiu classificar corretamente os 3 componentes do exemplo, pois a saída do crítico na Tabela 20 está fora do intervalo apresentado na última coluna desta mesma tabela.

4.4.4 Comitê de métodos (*ensemble*)

Com os métodos discutidos nas subseções anteriores, um comitê de métodos (*ensemble*) será montado, pois conforme descrito em [13] os métodos híbridos vêm sendo usados, com grande sucesso em diversas áreas, envolvendo sistemas de visão computacional. A Figura 48 exemplifica os passos do processo, utilizados nesse trabalho, para avaliar se um *bracket* possui defeito.

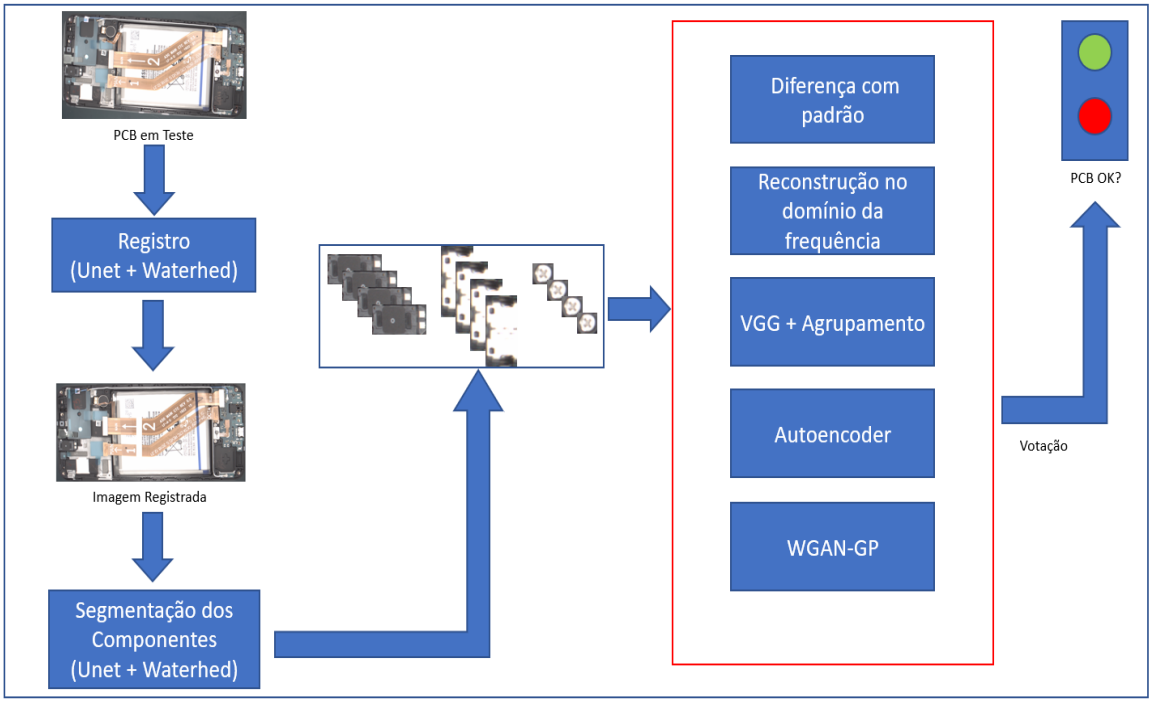


Figura 48 - Comitê de Métodos proposto

De modo a facilitar o entendimento, a metodologia utilizada pode ser resumida pelos seguintes passos:

- Imagens são obtidas na linha de produção, conforme apresentado na Figura 23;
- Manualmente os componentes a serem avaliados e o *bracket* são anotados utilizando a ferramenta Vott;
- As anotações Vott são exportadas em formato JSON a um software em *Python* que cria as máscaras para o treinamento da U-Net;
- Faz-se o treinamento da U-Net considerando um método de DA durante *n* épocas, tanto para o *bracket* como para os componentes;
- Utilizando-se da junção U-Net e *watershed* faz-se o registro dos *brackets* e, posteriormente, a segmentação dos componentes a serem testados;
- A partir dos componentes segmentados faz-se o treinamento, considerando *n* épocas, dos modelos CAE, WGAN-GP e Agrupamento baseado nas *features* extraídas pela VGG;
- Considerando os componentes segmentados e uma imagem de referência (*template*), calcula-se o limiar de para os métodos de

subtração a partir de imagem padrão e reconstrução baseada na 2D DFT;

- Considerando os componentes segmentados calcula-se o limiar de aprovação para o método VGG Agrupamento, através da distância entre as features do componente e o respectivo centroide do seu grupo;
- Ainda considerando os componentes segmentados, calcula-se o limiar de aprovação para os métodos CAE e WGAN-GP;
- No teste, a imagem é apresentada a junção U-Net e *watershed* cuja saída é o registro da imagem em teste. Com a imagem registrada são segmentados os componentes e apresentados a cada um dos métodos, sendo o resultado final a resposta, por votação, do comitê;
- Caso algum componente seja classificado como defeituoso o *bracket* é reprovado e deve ser separado para o posto de retrabalho.

5 Resultados e Discussões

Neste capítulo são apresentados os resultados para dois modelos de *bracket*. No primeiro modelo (Modelo A) são apresentadas as acurácias, na detecção de anomalias, para os 22 componentes analisados. Para o segundo modelo (Modelo B) é apresentado somente o valor do *IoU Score* para o registro do *bracket* e segmentação dos componentes, esta análise foi incluída para exemplificar a eficácia do método quando ocorre a inserção de um novo produto na linha de produção.

5.1 Modelo A.

Para o Modelo A foram avaliados 22 componentes, cujas as imagens são apresentadas na Tabela 23. Já a Figura 49 exibe os gráficos da função de custo e da acurácia para o treinamento da *U-Net*. Para obter tais gráficos foram executadas 200 épocas de treinamento juntamente com um total de 17 imagens, que sofreram um processo de DA (*Data Augmentation*). A Tabela 21 defini os parâmetros do DA e demais parâmetros do treinamento. Após o treinamento o valor da acurácia alcançado foi de 0,9879.

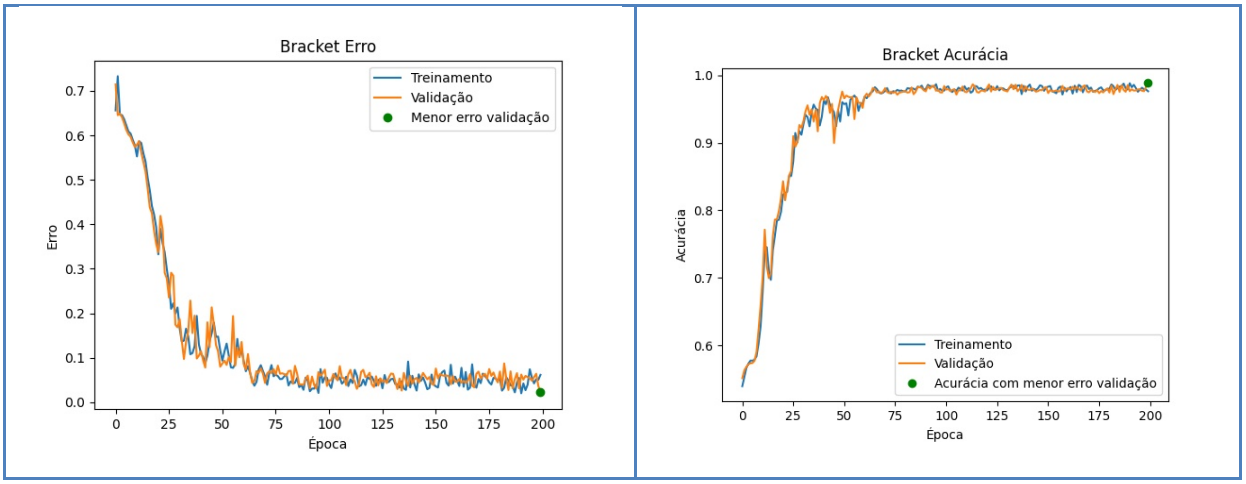


Figura 49 - Gráficos da Função de Erro e Acurácia da U-Net para o Modelo A.

O resultado do *IoU Score* considerando o registro do Modelo A é apresentado na Tabela 22. As 60 imagens de teste são diferentes das imagens utilizadas no treinamento, sendo que tais imagens foram geradas a partir das 17 imagens originais utilizando-se do DA apresentado na Tabela 21.

Tabela 21 - DA e Parâmetros do Treinamento para o Modelo A

Dados do DA (*Data Augmentation*)				
Imagens Originais	**Tamanho do *Batch***	**Rotação (Graus)**	**Translação Vertical (%)**	**Translação Horizontal (%)**
17	4	5	0,02	0,02
Dados do treino da U-Net				
Épocas	***Learn Rate***	**Otimizador**	***Loss***	***Métricas***
200	0,001	Adam	*Binary Crossentropy*	*Accuracy*
Resultado Treinamento				
Acurácia:	0,9879	**Tempo:**	18 m e 43 s.	

Tabela 22 - Resultados do Registro para o Modelo

Modelo	**Algoritmo**	**IoU Score**	***Tempo (s)***	***Imagens***
Modelo A	U-Net e *watershed*	0,9846 ±0,0027	0,4511 ±0,076	60

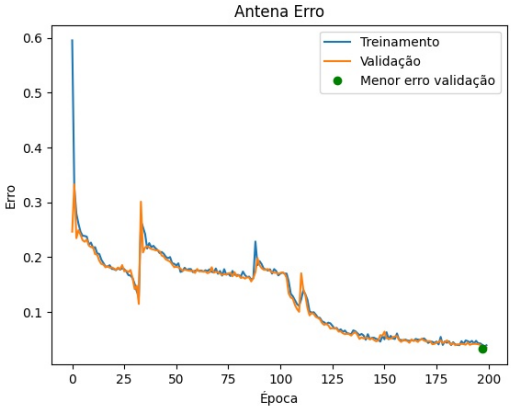
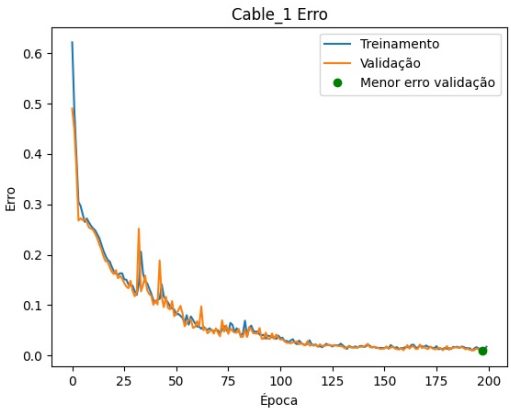
Com o registro da PCB, foi feito o treinamento de uma nova U-Net para cada componente a ser testado. Um fator de 1,5 vezes foi utilizado para determinar a área da imagem, ao redor do componente, a ser utilizada no treinamento. Deste modo, a região segmentada inicia-se no centro do componente e cresce 1,5 vezes o valor da largura do componente no eixo x e 1,5 vezes o valor da altura do componente no eixo y. O fator 1,5 foi definido de forma empírica, com esse valor conseguiu-se o melhor *IoU Score* que será apresentado posteriormente. A segmentação destas áreas é feita, de forma automática, pelo programa Python devolvido neste trabalho.

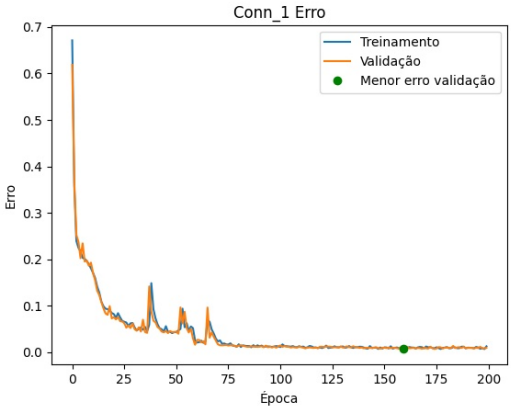
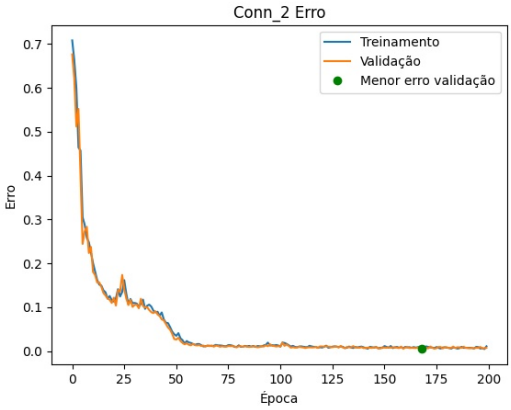
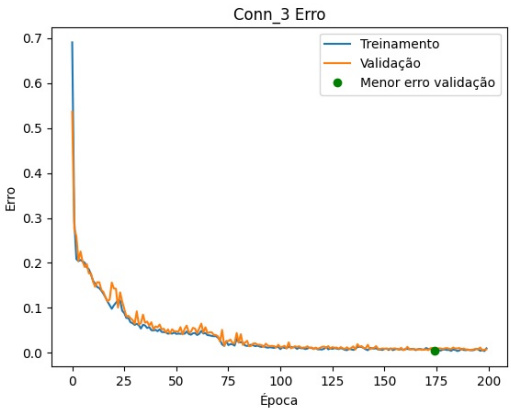
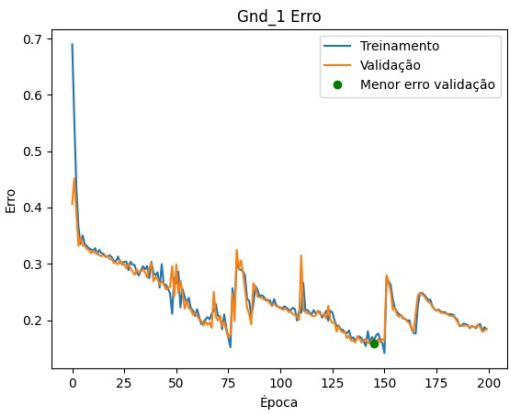
Os parâmetros utilizados no treinamento são praticamente os mesmos apresentados na Tabela 21. Somente os dados do DA foram alterados, uma vez que o *bracket* está devidamente registrado, não existe a possibilidade de rotação e translação dos componentes, sendo consideradas apenas diferenças de brilho (*brigtness_range*: [0,9;1,3]) para gerar as novas imagens. As alterações de brilho foram consideradas no DA, pois a imagem do *bracket* é capturada na borda de descida do sensor (Figura 23). Porém dependendo da posição do *bracket* sobre a

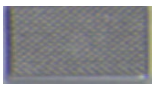
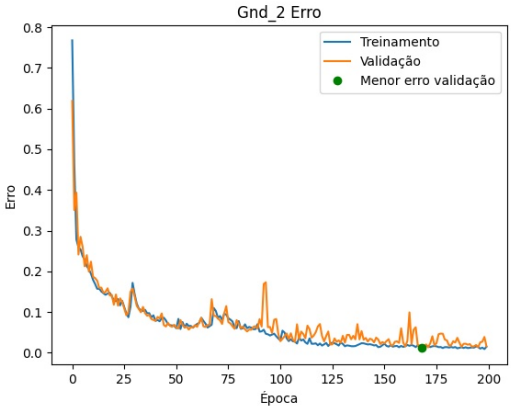
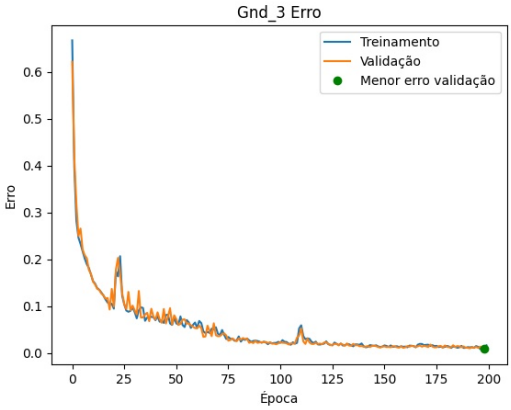
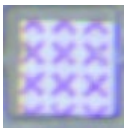
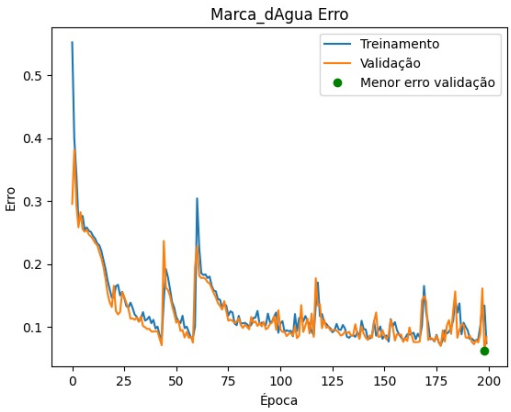
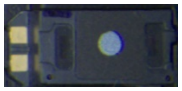
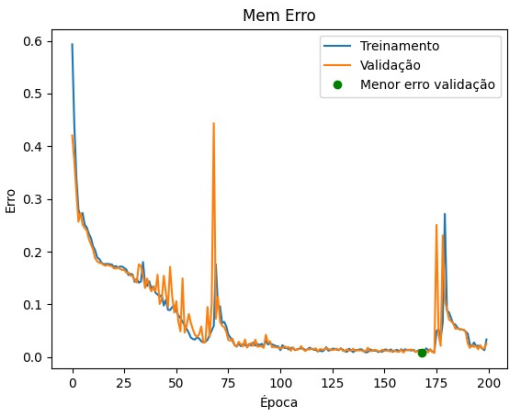
esteira, a aquisição pode ser acionada em um momento que o mesmo não esteja uniformemente iluminado.

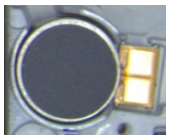
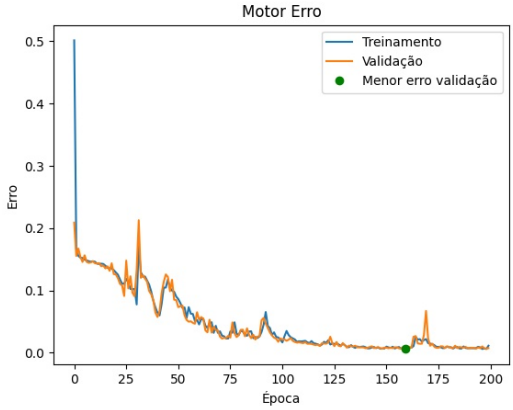
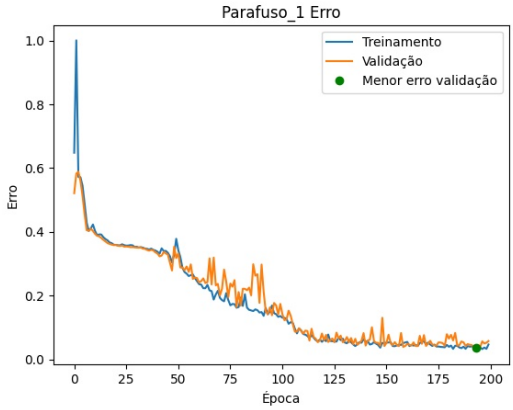
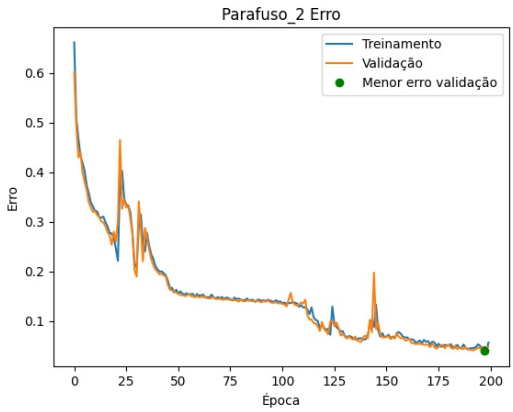
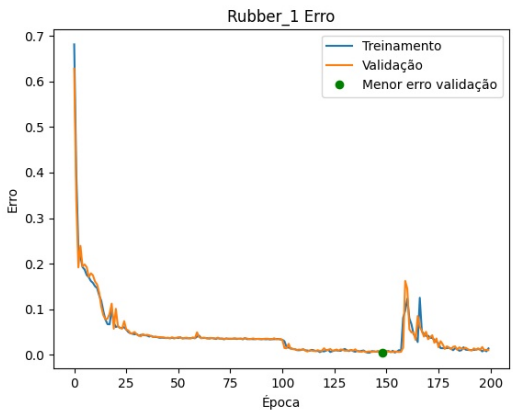
Os resultados de acurácia, gráficos da função de custo (considerando a função *Binary Crossentropy*) e *IoU Score* são apresentados na Tabela 23, pode-se observar que não houve *overfitting* no treinamento. O tempo médio para o treinamento de um componente foi de 269,14 s (4 m e 28 s) e o tempo total de treinamento foi de 5921,08 s (aproximadamente 1 h e 38 m).

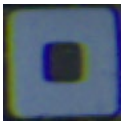
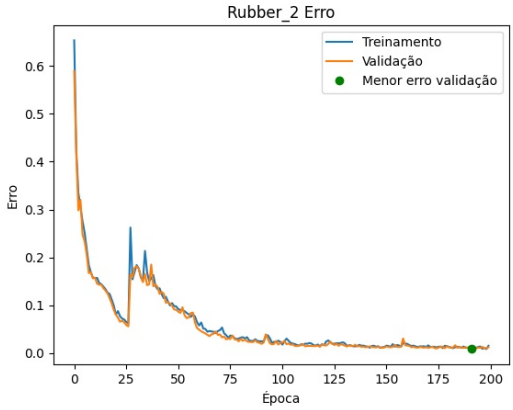
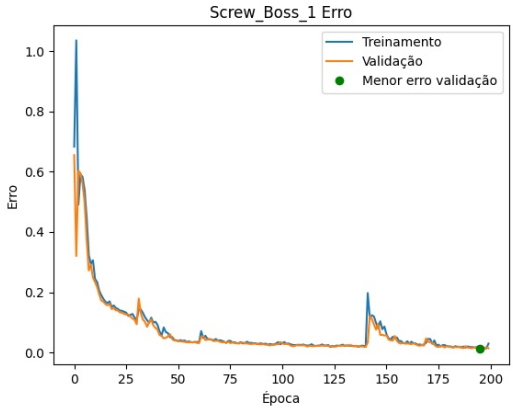
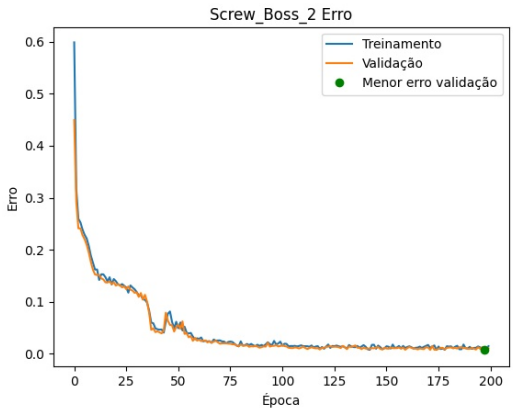
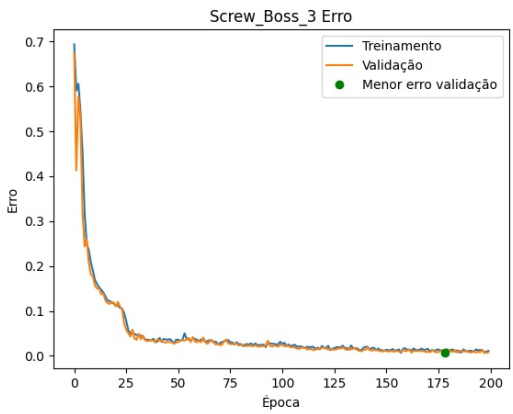
Tabela 23 - Acurácia e Função de Custo do treinamento U-Net para os componentes do Modelo A.

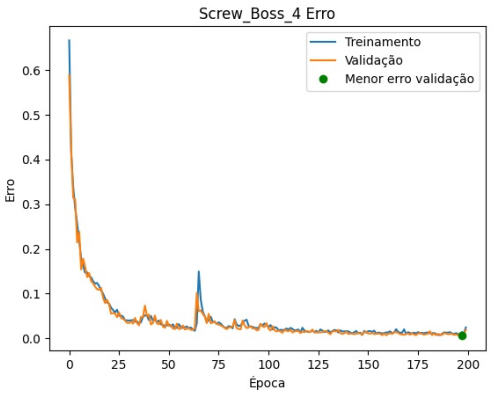
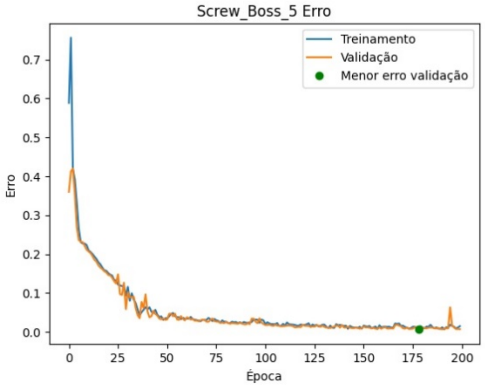
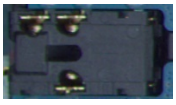
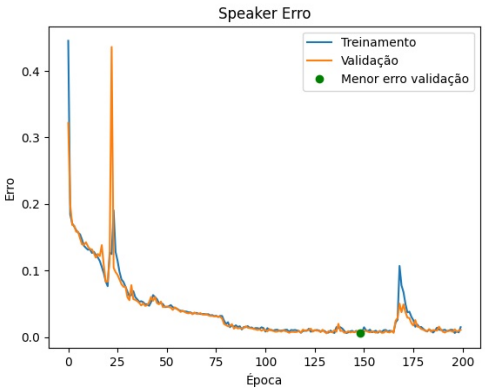
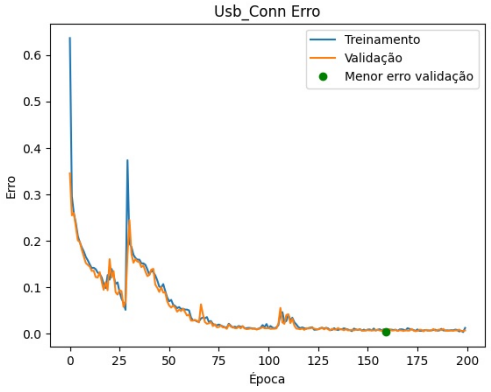
Componente	Gráfico (Função de Custo)	Acurácia	IoU Score
Antena 		0,9708	0,7659 ±0,0876
Cable_1 		0,9860	0,8030 ±0,0828

Conn_1 		0,9810	0,8852 ±0,0370
Conn_2 		0,9913	0,9095 ±0,0344
Conn_3 		0,9901	0,8704 ±0,0504
Gnd_1 		0,9331	0,3820 ±0,1207

Gnd_2 		0,9876	0,6423 ±0,1354
Gnd_3 		0,9833	0,7391 ±0,1889
Marca_dAgua 		0,9851	0,7146 ±0,1140
Mem 		0,9900	0,9200 ±0,0419

Motor 		0,9905	0,8266 ±0,0742
Parafuso_1 		0,9297	0,7418 ±0,1429
Parafuso_2 		0,9750	0,8465 ±0,0632
Rubber_1 		0,9888	0,8888 ±0,0355

Rubber_2 		0,9846	0,7063 ±0,3189
Screw_Boss_ 1 		0,9843	0,8304 ±0,0693
Screw_Boss_ 2 		0,9858	0,7248 ±0,1858
Screw_Boss_ 3 		0,9854	0,7239 ±0,1175

Screw_Boss_4 		0,9860	0,6835 ±0,1832
Screw_Boss_5 		0,9814	0,7301 ±0,1969
Speaker 		0,9896	0,9111 ±0,0456
Usb_Conn 		0,9880	0,8919 ±0,0323
Tempo:	1 hora e 38 minutos		

A diferença entre a acurácia e o *IoU Score* apresentado na Tabela 23 pode ser explicada pelo processo de criação das máscaras para o cálculo do *IoU Score.* Tanto nas imagens marcadas manualmente no *Vott* quanto para a máscara calculada na predição, considera-se um retângulo sem rotação ao redor do maior contorno na máscara, sendo este criado pela função *boundingRect* presente na biblioteca *opencv.* Assim, um ponto que foi marcado incorretamente no *Vott* pode prejudicar o cálculo do *IoU Score.*

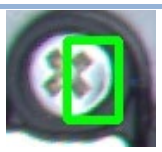
Os valores de limiar, utilizados na binarização das predições oriundas da U-Net, também podem causar a anomalia em discussão. Na saída da U-Net tem-se uma função sigmoide que retorna valores no intervalo de [0,1], na binarização está sendo considerado como região do componente todo valor maior ou igual a 0,5. Caso outros valores sejam considerados na formação das máscaras, os valores de acurácia e *IoU Score* podem tornar-se semelhantes.

A configuração do DA também interfere no treinamento da U-Net, sendo que para alguns componentes a alteração no brilho prejudicou a segmentação. O componente Gnd_1 é um exemplo desse caso, removendo-se a configuração de brilho, foi possível um melhor resultado de IoU Score conforme apresentado na Tabela 24.

Para alguns componentes percebeu-se que a combinação U-Net e watershed melhora a segmentação. Com o uso do watershed, os erros da seleção manual dos componentes na ferramenta Vott é corrigido, este fato pode ser percebido analisando os retângulos criados nas imagens. Embora em algumas situações o IoU Score seja menor, visualmente a seleção está mais adequada. Dentre os componentes apresentados na Tabela 23, utilizou este recurso nos seguintes componentes: Parafuso_1, Parafuso_2, Gnd_1 e Gnd_2. A Tabela 24 exemplifica a diferença na segmentação utilizando os dois métodos propostos, além de mostrar os ganhos com as modificações executadas.

Todos os métodos a serem utilizados na avaliação requerem um treinamento baseado em componentes sem defeito, onde são calculadas a média e desvio padrão em relação à saída de cada algoritmo. Deste modo, as 17 imagens originais tiveram os componentes segmentados. Após a segmentação os componentes foram avaliados manualmente excluindo-se as imagens cuja segmentação foi incorreta, formando assim base de treinamento para cálculo do limiar de aprovação.

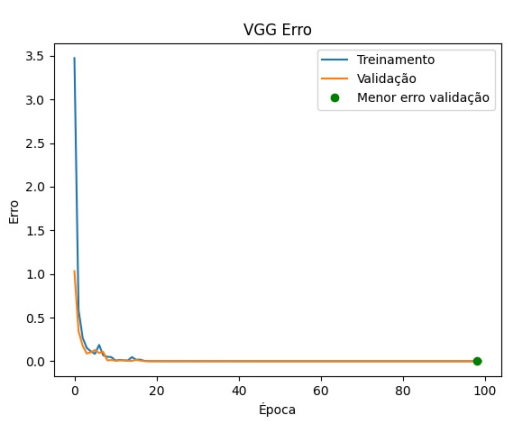
Tabela 24 - Diferença na segmentação usando U-Net e U-Net combinada com watershed.

Componente	U-Net	U-Net e Watershed	Erosão/ Dilatação	Iou Score
Parafuso_1			(40,40) (40,40)	0,6328 (0,2391)
Parafuso_2			(40,40) (40,40)	0,7190 (0,0846)
Gnd_1			(30,30) (30,30)	0,7141 (0,1246)
Gnd_2			(30,30) (40,40)	0,8342 (0,1265)

A Tabela 25 apresenta os dados para o treinamento da VGG considerando o TL, além disso, também é apresentado o resultado do treinamento considerando 17 classes de componentes. Os componentes parafusos e screw bosses foram agrupados em 2 classes, deste modo o número de classes é diferente do número de componentes a serem avaliados. Após o treinamento e remoção da camada Softmax, e utilizando as imagens de componentes sem defeito, foi feito o treinamento do K-means, obtendo-se como resultado um coeficiente de silhueta de 0,4624.

A Tabela 26 apresenta os gráficos de *Z-Score* considerando a distância entre a imagem usado no treinamento e seu respectivo centroide. O eixo vertical (*index*) do gráfico representa o índice da amostra utilizada no cálculo do *Z-Score,* sendo que a primeira amostra possui índice 0. Esses gráficos serão úteis para determinar o número de desvios padrões que irão compor o limiar de aprovação.

Tabela 25 - Treinamento VGG com transfer learning

<table>
<tr><td colspan="2">Imagens Originais (para cada componente)</td><td>17</td></tr>
<tr><td>Modificação na VGG</td><td colspan="2">• Removida as camadas conectadas originais.
• Acrescentado duas camadas conectadas com 1000 neurônios cada e ativação ReLU.
• Acrescentado uma camada Softmax para classificar 17 componentes.</td></tr>
<tr><td colspan="3">Dados do DA</td></tr>
<tr><td>brigtness_range</td><td colspan="2">[0,9;1,3]</td></tr>
<tr><td colspan="3">Treinamento da Rede Neural</td></tr>
<tr><td>Função de Custo</td><td>Otimizador</td><td>Métrica</td></tr>
<tr><td>categorical_crossentropy</td><td>Adam</td><td>accuracy</td></tr>
<tr><td colspan="3">Resultado para 200 épocas</td></tr>
<tr><td colspan="3"> 

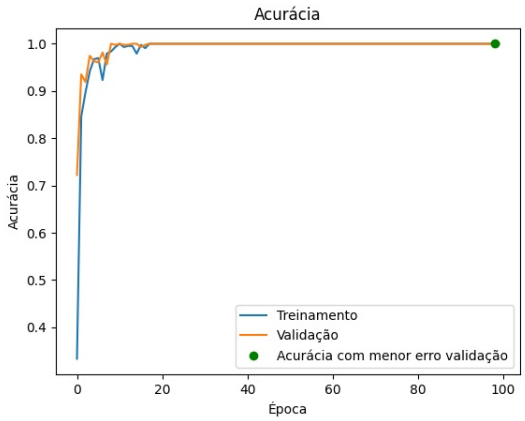
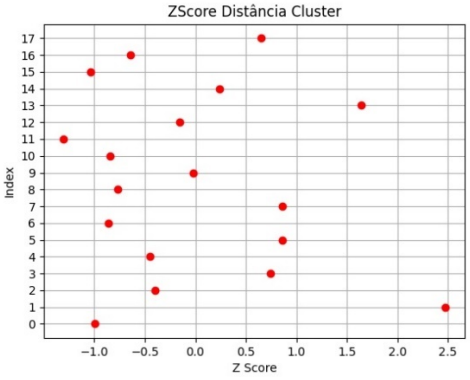
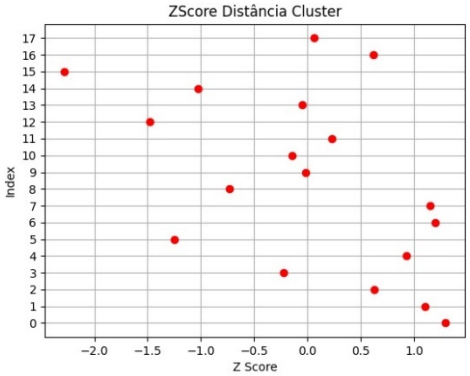
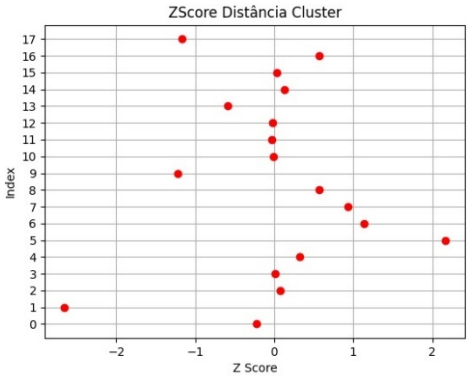
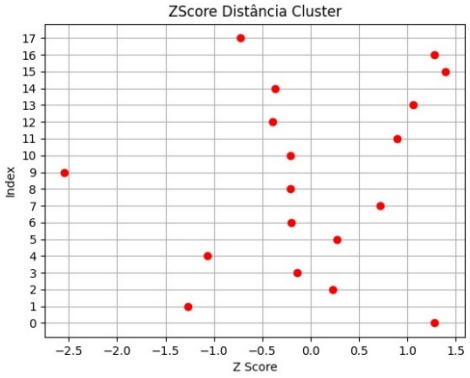
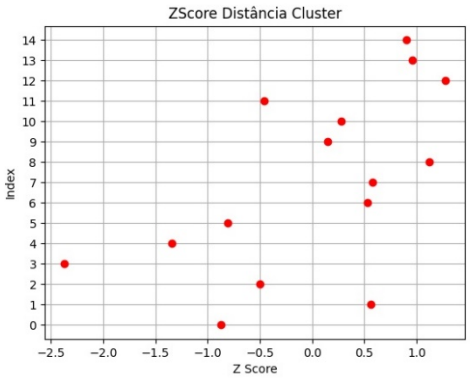
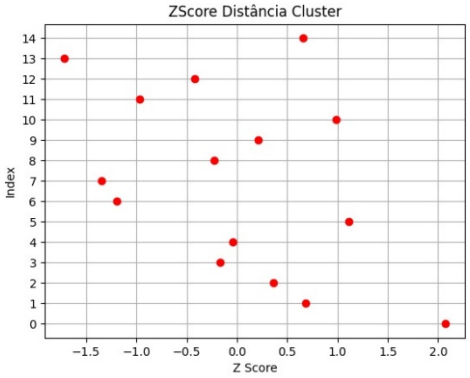
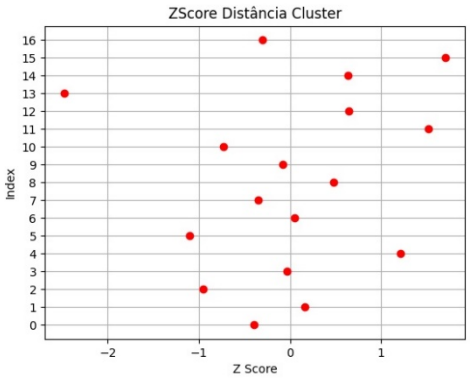
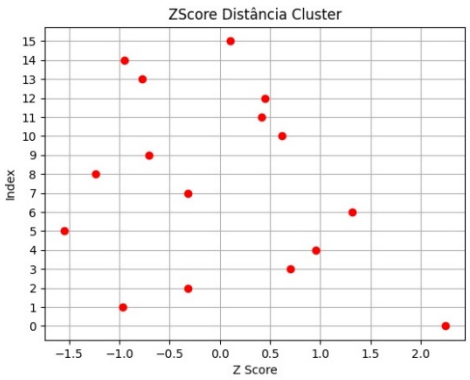
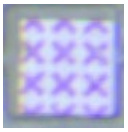
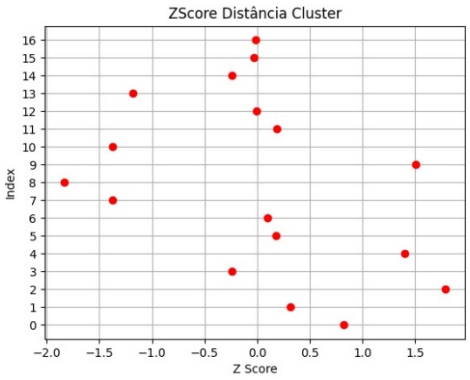
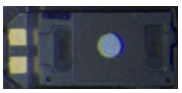
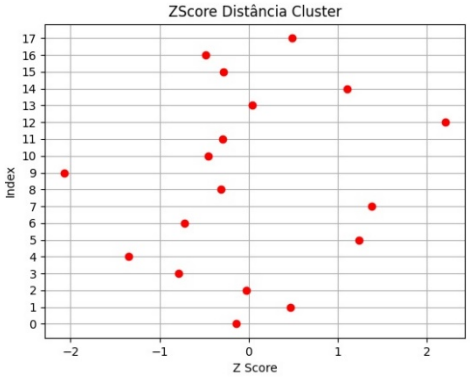
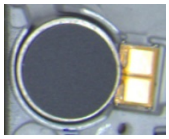
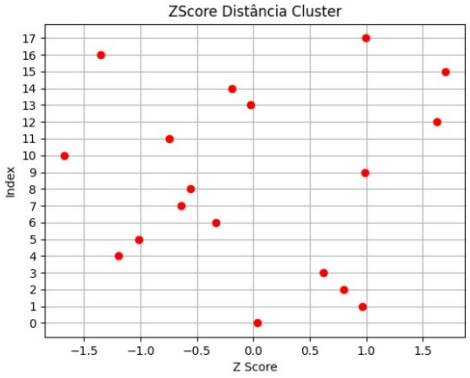
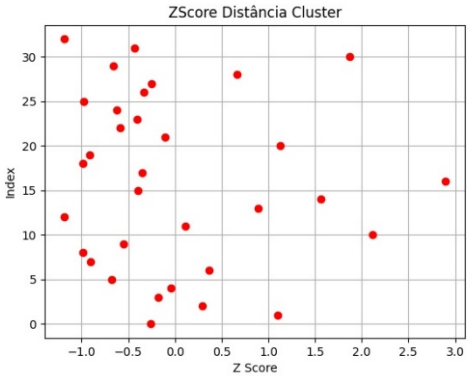
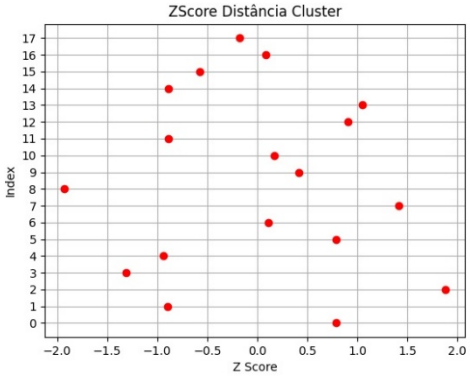
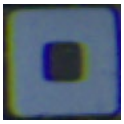
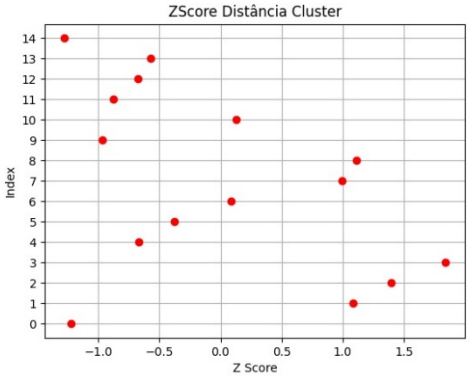
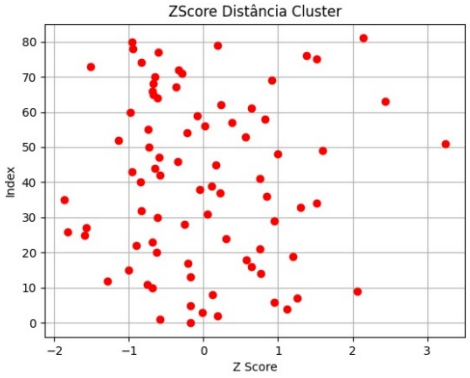
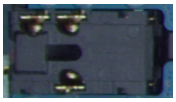
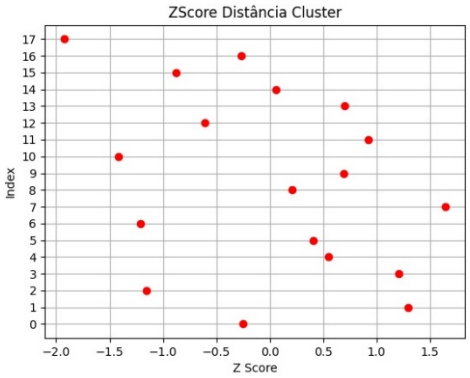

</td></tr>
<tr><td colspan="2">Tempo de Treinamento da VGG:</td><td>435 s (7 m e 15 s).</td></tr>
<tr><td colspan="2">Tempo de Treinamento Kmeans:</td><td>22 s.</td></tr>
</table>

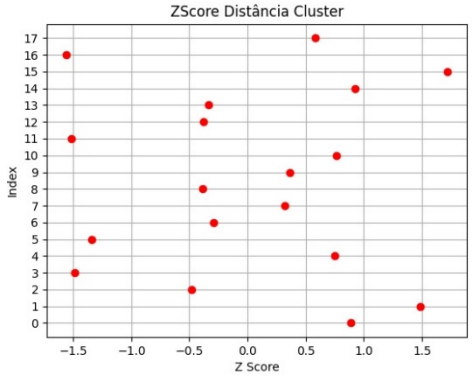
Tabela 26 - Z-Score Distância Centróide

Componente	*Z-Score*
Antena 	
Cable_1 	
Conn_1 	
Conn_2 	

Conn_3 	
Gnd_1 	
Gnd_2 	
Gnd_3 	

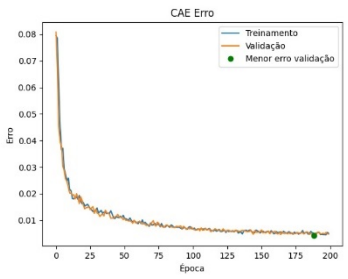
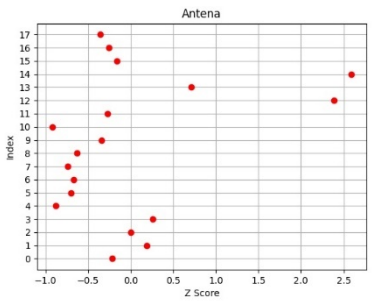
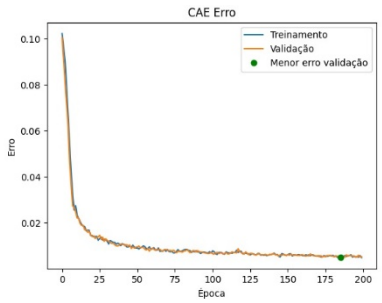
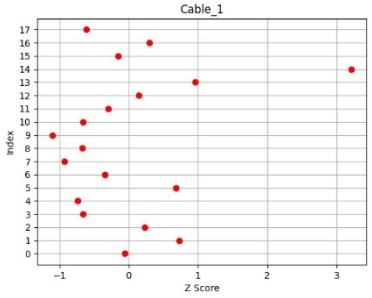
Marca_dAgua 	
Mem 	
Motor 	
Parafuso 	

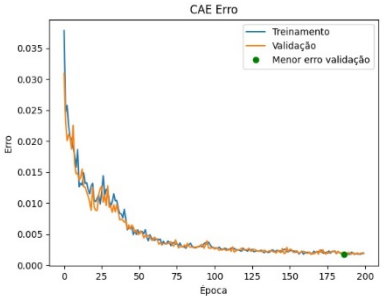
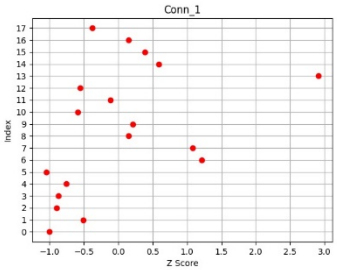
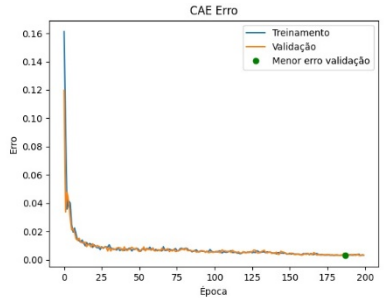
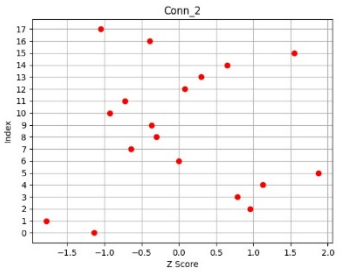
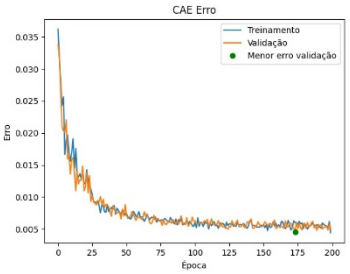
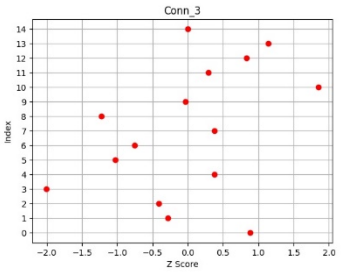
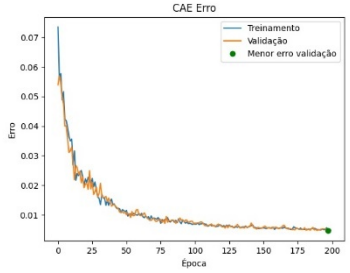
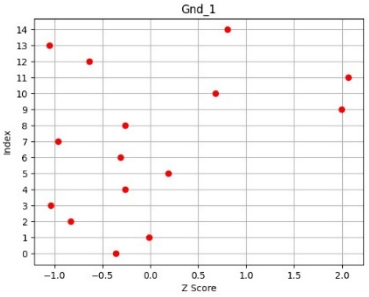
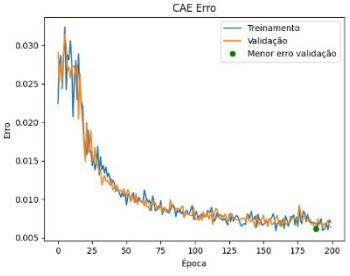
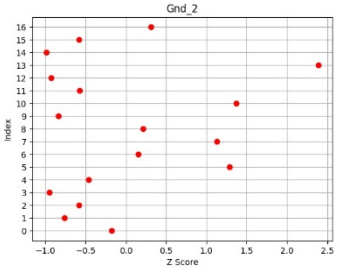
Rubber_1 	
Rubber_2 	
Screw_Boss 	
Speaker 	

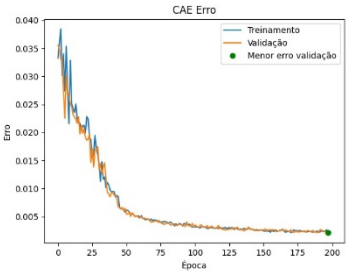
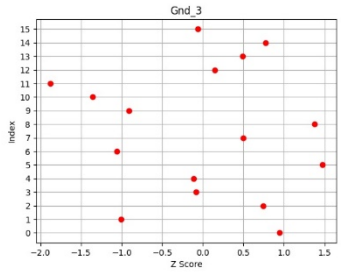
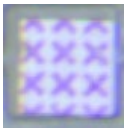
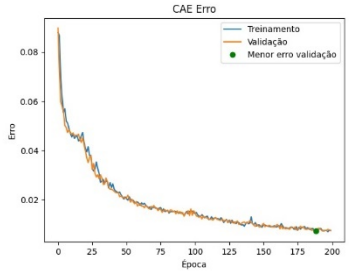
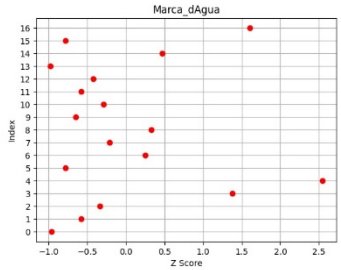
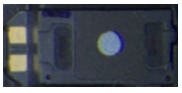
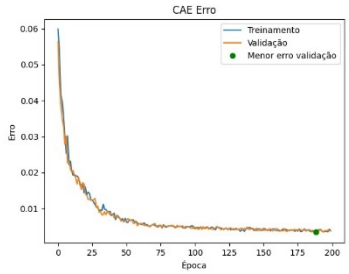
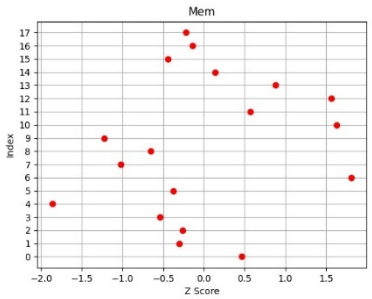
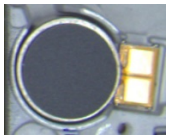
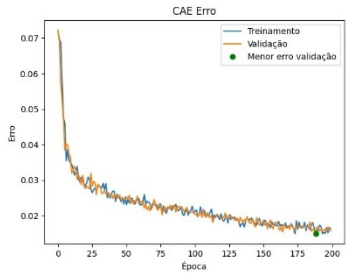
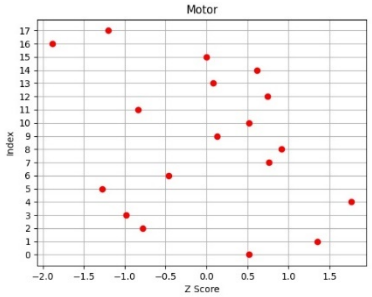
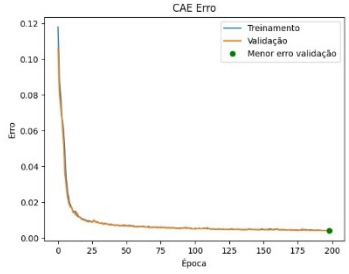
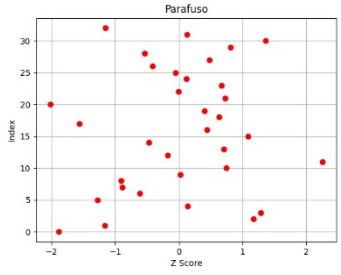
Usb_Conn 	

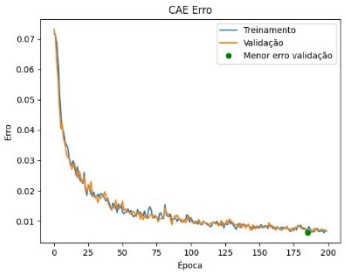
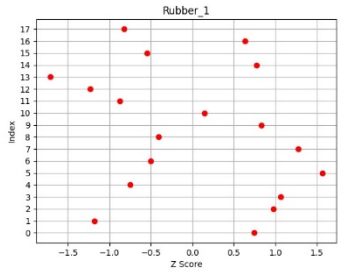
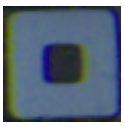
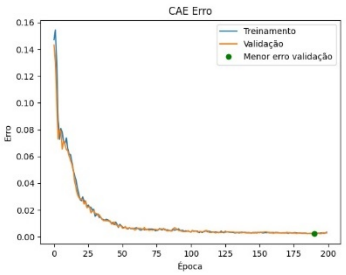
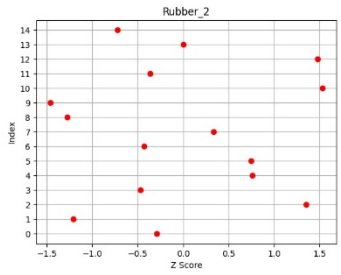
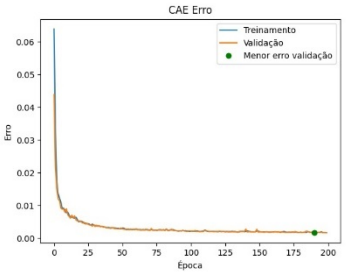
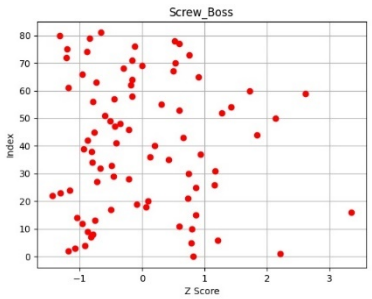
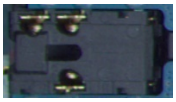
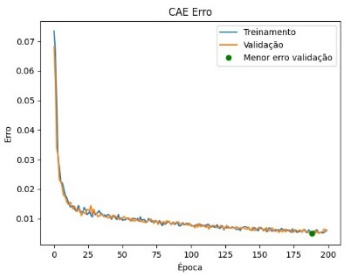
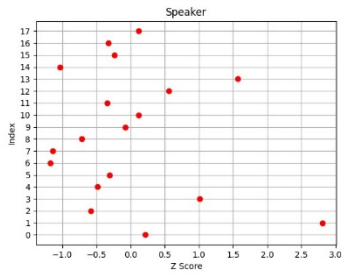
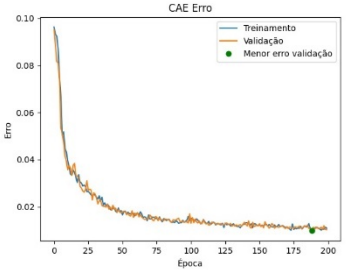
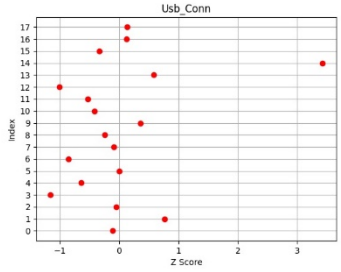
A Tabela 27 apresenta os resultados do treinamento do autoencoder considerando 200 épocas. Novamente utilizou-se o DA apresentado na Tabela 25, o tempo total de treinamento foi 394,4472 s (6 m e 34 s). O *Z-Score,* considerando o MSE, calculado pela reconstrução do autoencoder e a imagem original também é apresentado.

Tabela 27 - Resultado do Treinamento Autoencoder.

Componente	Função de Custo	*Z-Score*
Antena 		
Cable_1 		

Conn_1 		
Conn_2 		
Conn_3 		
Gnd_1 		
Gnd_2 		

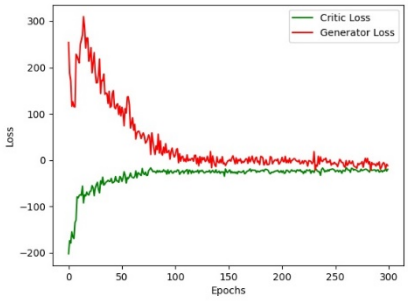
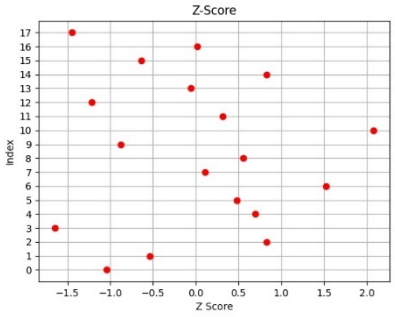
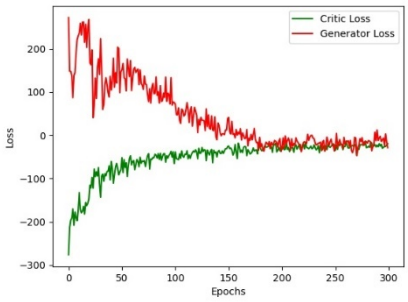
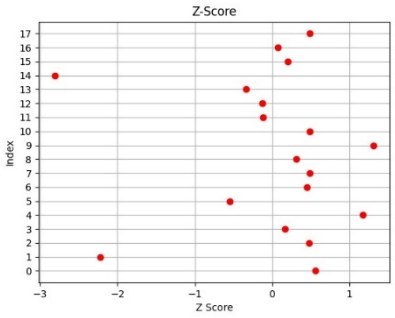
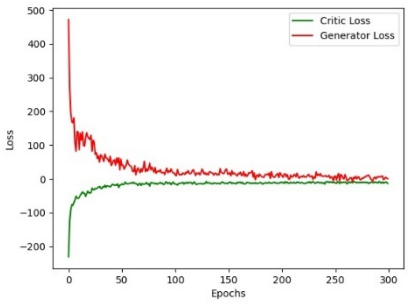
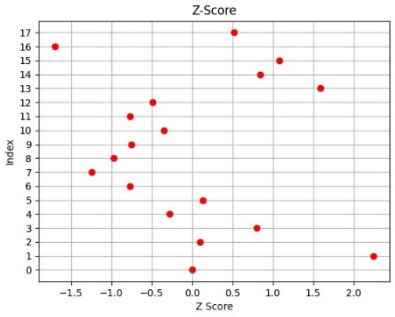
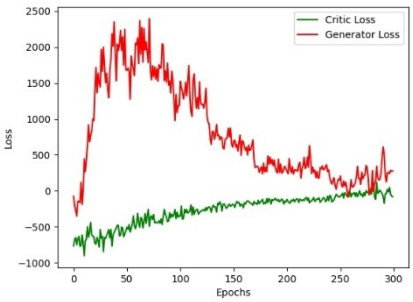
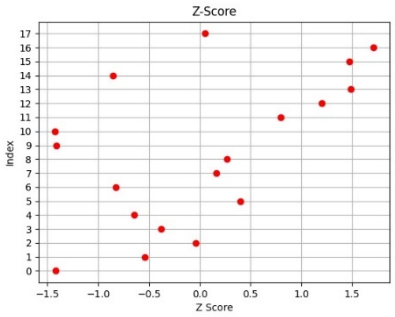
Gnd_3 		
Marca_dAgua 		
Mem 		
Motor 		
Parafuso 		

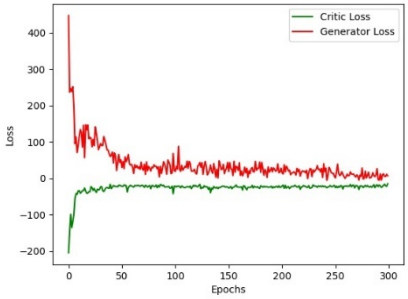
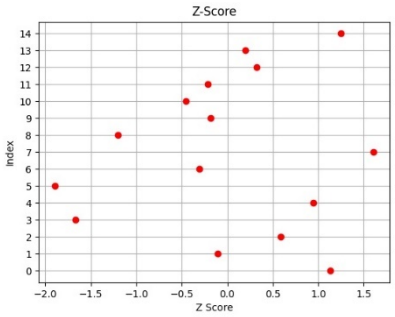
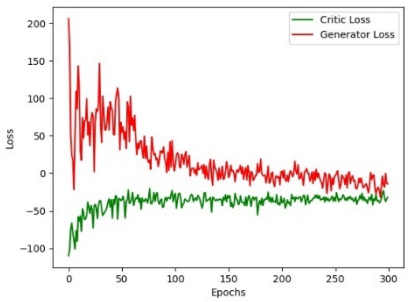
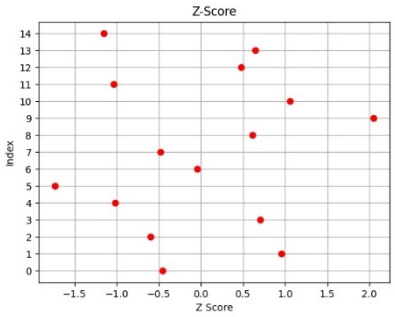
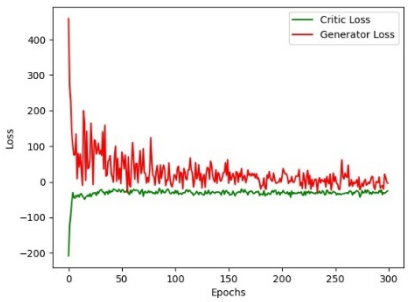
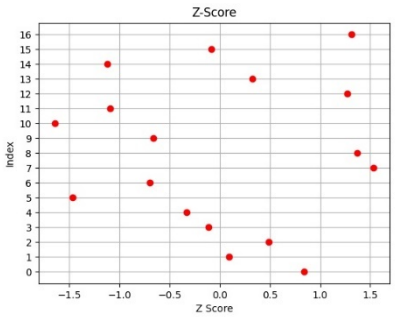
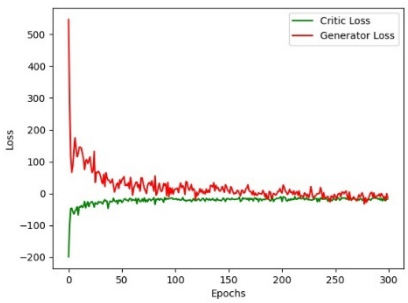
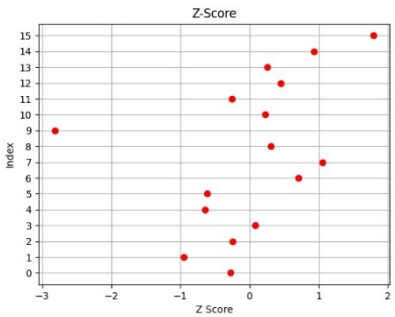
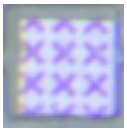
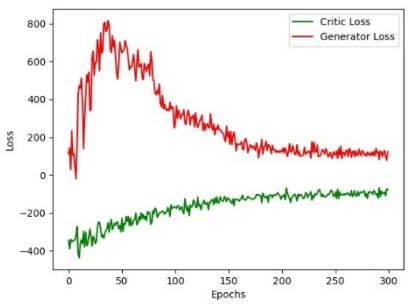
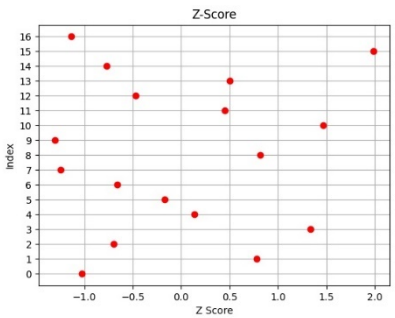
Rubber_1 		
Rubber_2 		
Screw_Boss 		
Speaker 		
Usb_Conn 		
Tempo:	6 m e 34 s.	

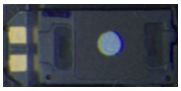
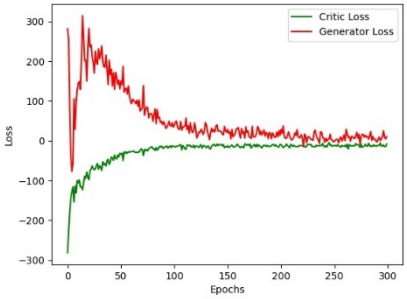
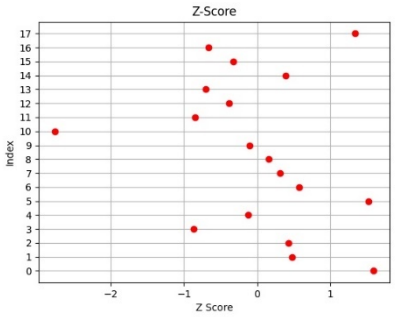
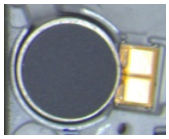
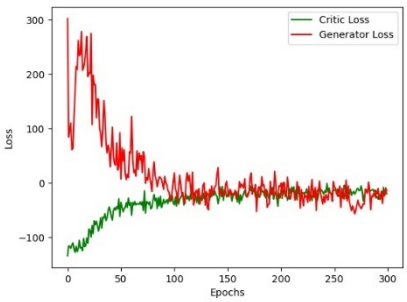
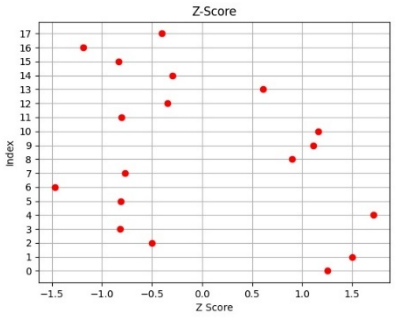
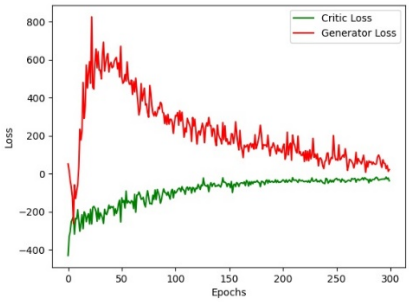
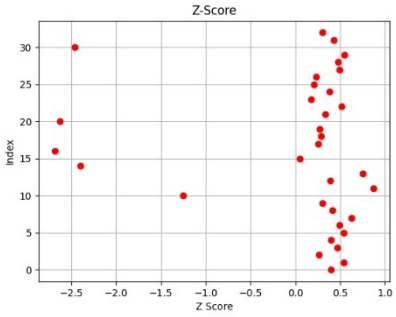
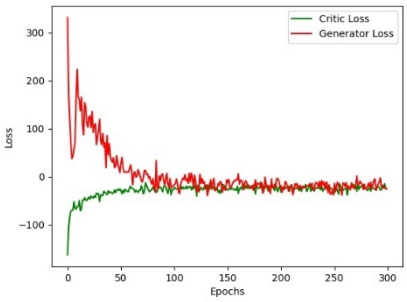
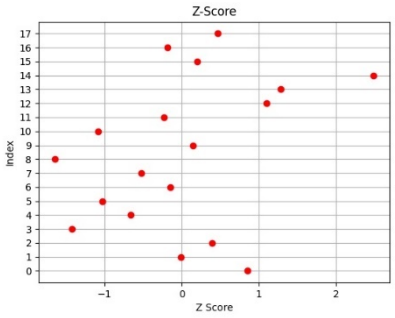
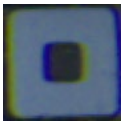
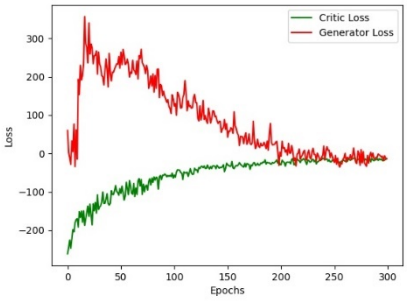
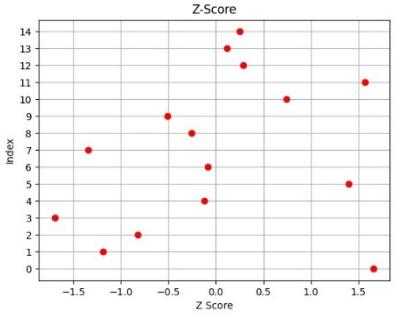
A Tabela 28 traz os resultados do treinamento da WGAN-GP para cada um dos componentes considerando 300 épocas, além disso, são apresentados os gráficos

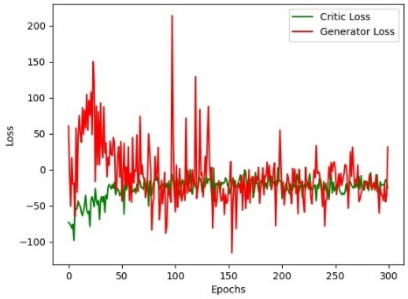
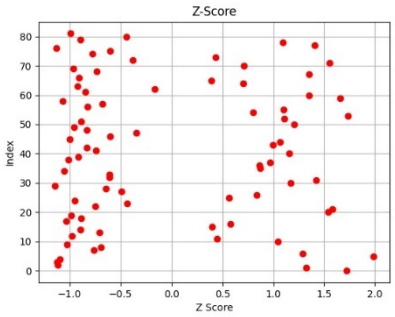
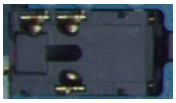
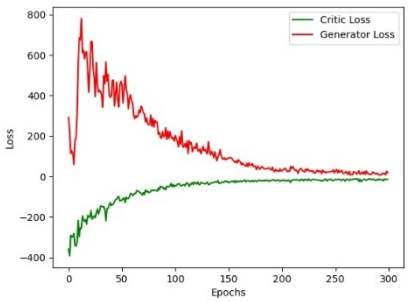
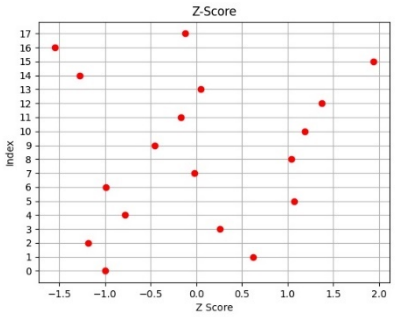
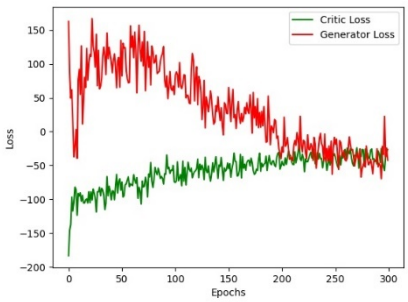
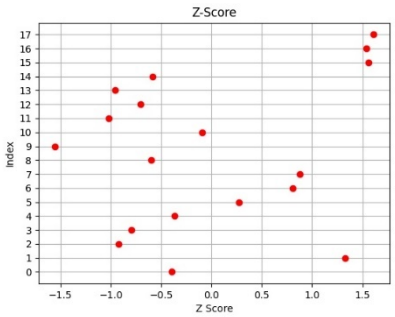
Z-Score, considerando a saída do crítico e o tempo total de treinamento. Novamente utilizou-se o DA apresentado na Tabela 25.

Tabela 28 - Resultado do treinamento da GAN

Componente	Função de Custo	*Z-Score*
Antena 		
Cable_1 		
Conn_1 		
Conn_2 		

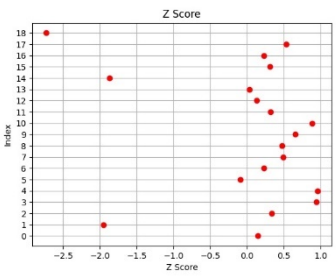
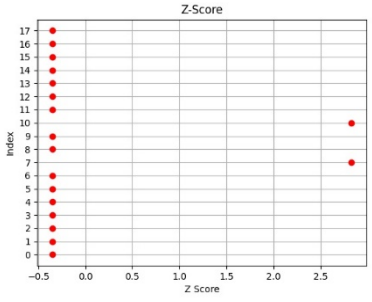
Conn_3 		
Gnd_1 		
Gnd_2 		
Gnd_3 		
Marca_dAgua 		

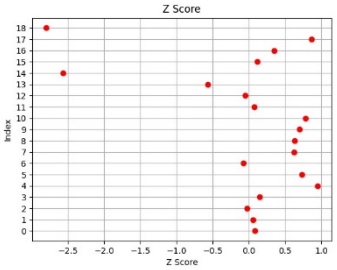
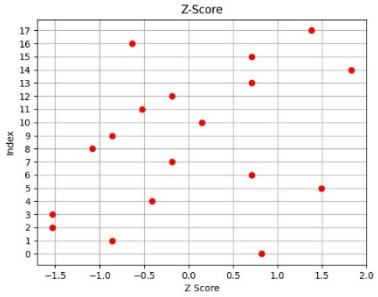
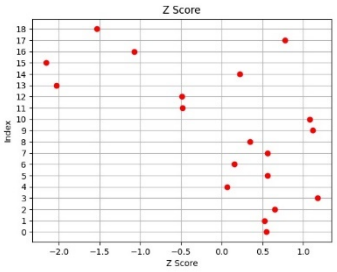
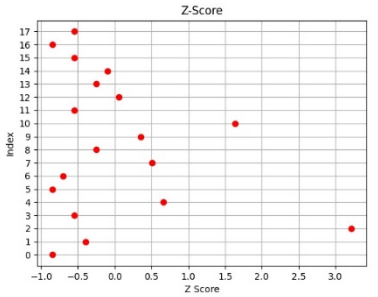
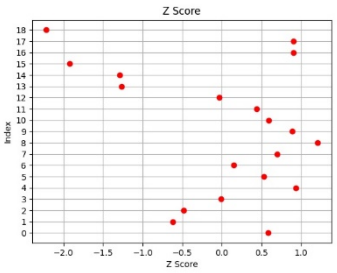
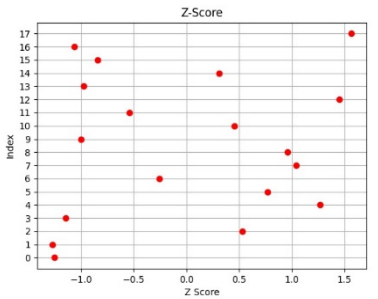
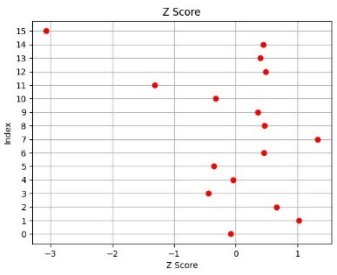
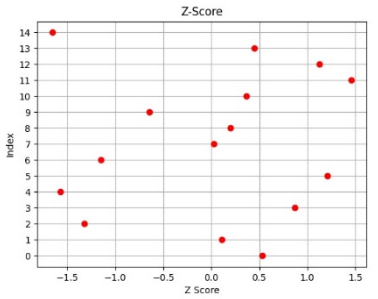
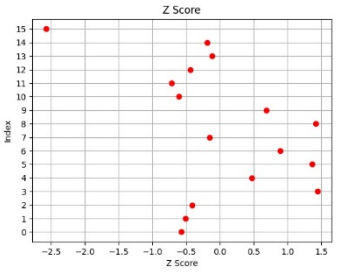
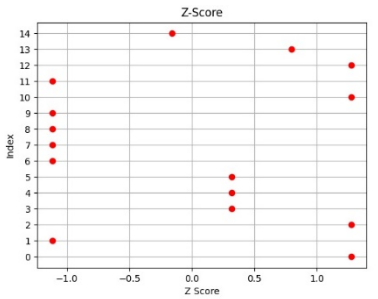
Mem 		
Motor 		
Parafuso 		
Rubber_1 		
Rubber_2 		

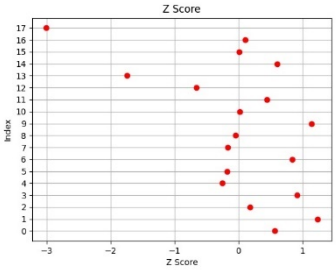
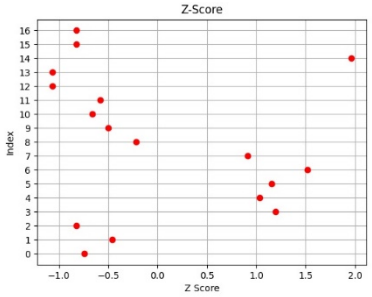
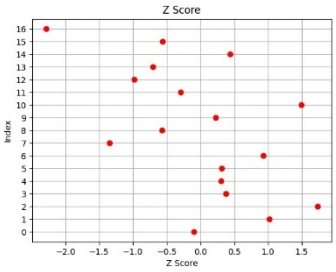
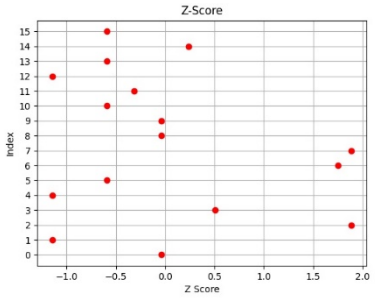
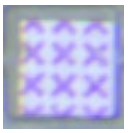
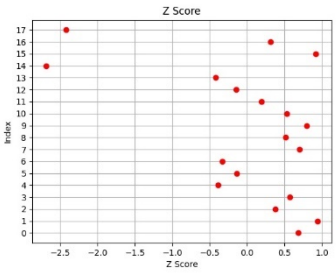
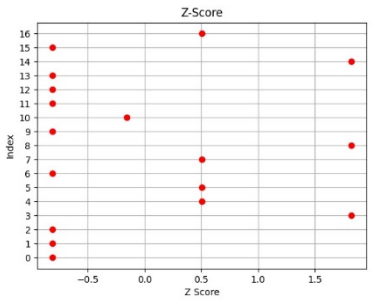
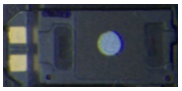
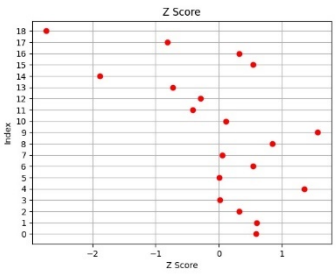
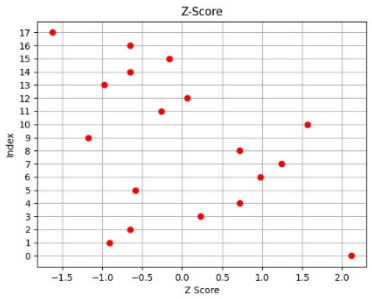
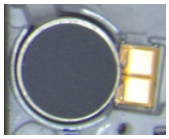
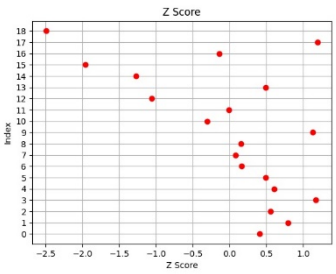
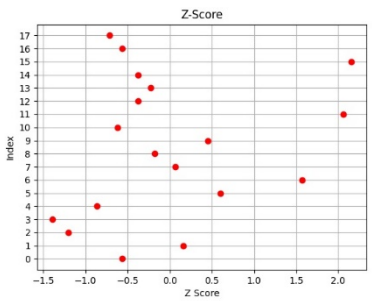
Screw_Boss 		
Speaker 		
Usb_Conn 		
Tempo:	17 h e 43 m	

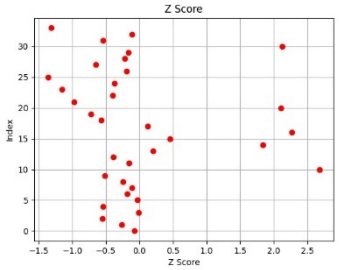
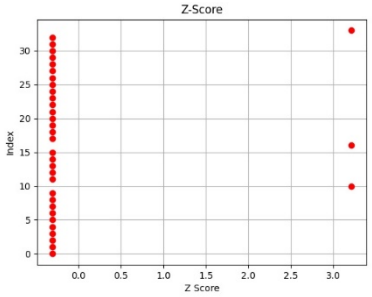
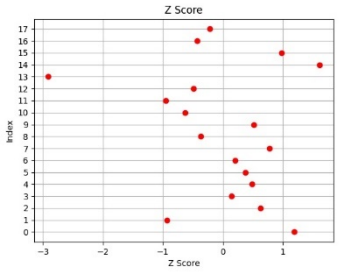
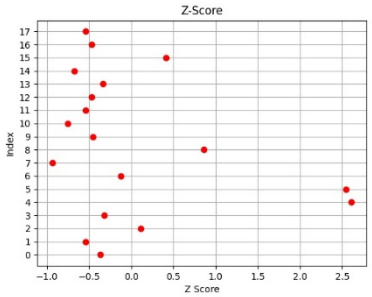
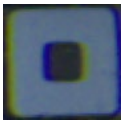
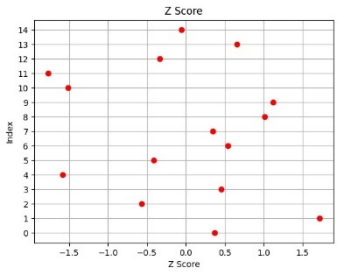
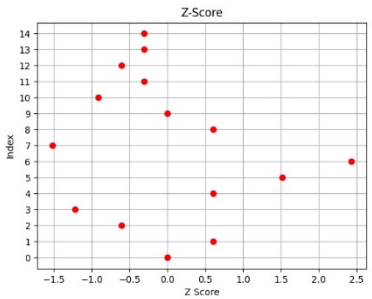
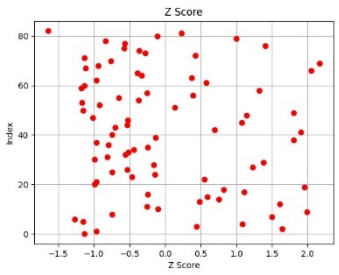
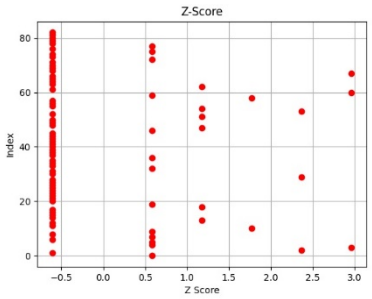
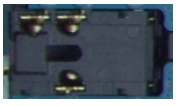
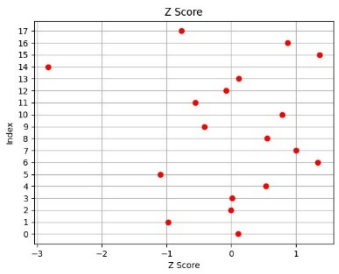
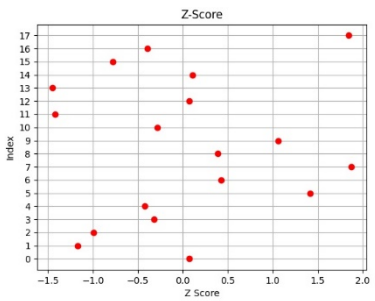
A Tabela 29 exibe o gráfico do *Z-Score* para os métodos: subtração baseado em *template* e reconstrução usando a 2D DFT.

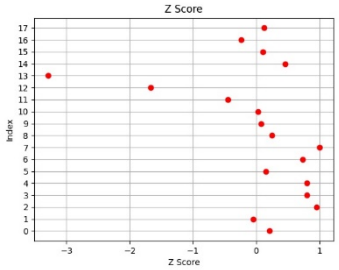
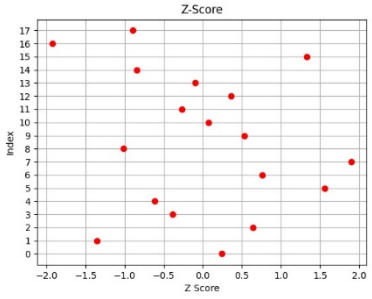
Tabela 29 - Z-Scores Subtração e 2D DFT

Componente	***Z-Score*** **Subtração *Template***	***Z-Score*** **Reconstrução 2D DFT**
Antena 		

Cable_1 		
Conn_1 		
Conn_2 		
Conn_3 		
Gnd_1 		

Gnd_2 		
Gnd_3 		
Marca_dAgua 		
Mem 		
Motor 		

Parafuso 		
Rubber_1 		
Rubber_2 		
Screw_Boss 		
Speaker 		

Usb_Conn		
		

Para o teste do comitê, utilizou-se o processo de DA para criar o conjunto de imagens de teste. Os parâmetros do DA são os mesmos apresentados na Tabela 21, acrescido da configuração de brilho utilizada no treinamento. Assim, as 17 imagens, sem defeito, deram origem a 100 imagens sem defeito. E as 18 imagens com defeitos deram origem à 10 imagens com defeito para cada componente, exceto para os componentes *Screw_Bosses* e Parafuso_1, pois os mesmos não possuíam nenhum exemplo de defeito no conjunto de imagens originais.

A Tabela 30 exibe o resultado da inspeção para o Modelo A, contendo os resultados de cada algoritmo e o resultado do comitê. O limiar de aprovação foi calculado considerando 2 vezes o desvio padrão a partir da média. Chegou-se a esse valor analisando os gráficos de *Z-Score*, a maioria dos valores estava presente nessa faixa.

Ainda considerando a tabela, são apresentados os resultados das predições verdadeiras negativas (TN – *True Negative*) e verdadeiras positivas (TP – *True Positive*). Visto que o objetivo do método é encontrar defeitos nos *brackets*, considerou-se como TP quando é encontrado um defeito no componente e a imagem foi previamente classificada como defeituosa.

Tabela 30 - Resultado Comitê para o Modelo A

Componente		Subtração	Subtração FFT	VGG Cluster	CAE	WGAN-GP	Comitê
Antena	TP	0	0	0	2	7	0
	TN	100	99	97	100	23	100
	Acurácia	0,9090	0,9000	0,8818	0,9272	0,2727	0,9090
Cable_1	TP	10	9	0	0	0	0
	TN	100	100	100	93	86	100
	Acurácia	1,0000	0,9909	0,9090	0,8454	0,7818	0,9090

Conn_1	TP	10	10	2	8	4	9
	TN	100	16	68	87	74	86
	Acurácia	1,0000	0,2363	0,6363	0,8636	0,7090	0,8636
Conn_2	TP	10	0	5	10	6	6
	TN	98	100	99	81	92	100
	Acurácia	0,9818	0,9090	0,9454	0,8272	0,8909	0,9636
Conn_3	TP	2	3	5	10	10	7
	TN	100	0	61	23	0	23
	Acurácia	0,9227	0,0272	0,600	0,0300	0,09090	0,2727
Gnd_1	TP	8	7	9	7	8	7
	TN	78	76	7	67	6	66
	Acurácia	0,7818	0,7545	0,1454	0,6727	0,1272	0,6636
Gnd_2	TP	0	2	0	6	10	0
	TN	100	31	100	80	69	94
	Acurácia	0,9090	0,300	0,9090	0,7818	0,7118	0,8545
Gnd_3	TP	0	4	9	10	10	9
	TN	92	42	85	96	55	89
	Acurácia	0,8363	0,4181	0,8545	0,9636	0,5909	0,8909
Marca_dAgua	TP	10	3	0	0	**10**	3
	TN	100	100	**94**	**94**	**100**	**100**
	Acurácia	1,0000	0,9363	0,8545	0,8545	1,0000	0,9363
Mem	TP	4	0	10	10	10	10
	TN	100	0	55	100	100	100
	Acurácia	0,9454	0,0000	0,5909	1,0000	1,0000	1,0000
Motor	TP	10	8	5	8	10	10
	TN	100	100	100	93	94	100
	Acurácia	1,0000	0,9818	0,9545	0,9181	0,9454	1,0000
Parafuso_1	TP	-	-	-	-	-	-
	TN	93	94	94	88	87	93
	Acurácia	0,9300	0,9400	0,9400	0,8800	0,8700	0,9300
Parafuso_2	TP	10	10	10	10	10	10
	TN	97	100	100	98	87	99
	Acurácia	0,9727	1,0000	1,0000	0,9818	0,8818	0,9909
Rubber_1	TP	10	0	0	0	0	0
	TN	100	100	100	100	98	100
	Acurácia	1,0000	0,9090	0,9090	0,9090	0,5909	0,9090
Rubber_2	TP	5	0	0	4	5	0
	TN	**100**	**100**	**100**	**97**	**76**	**100**

	Acurácia	0,9554	0,9090	0,9090	0,9181	0,7363	0,9090
Screw_Boss_1	TP	-	-	-	-	-	-
	TN	99	91	99	96	99	99
	Acurácia	0,9900	0,9100	0,9900	0,9600	0,9900	0,9900
Screw_Boss_2	TP	-	-	-	-	-	-
	TN	100	95	85	90	100	100
	Acurácia	1,0000	0,9500	0,8500	0,9000	1,0000	1,0000
Screw_Boss_3	TP	-	-	-	-	-	-
	TN	100	38	94	100	100	100
	Acurácia	1,0000	0,3800	0,9400	1,0000	1,0000	1,0000
Screw_Boss_4	TP	-	-	-	-	-	-
	TN	100	40	100	100	100	100
	Acurácia	1,0000	0,4000	1,0000	1,0000	1,0000	1,0000
Screw_Boss_5	TP	-	-	-	-	-	-
	TN	92	29	94	100	100	100
	Acurácia	0,9200	0,2900	0,9400	1,0000	1,0000	1,0000
Speaker	TP	10	10	0	10	10	10
	TN	100	0	100	100	89	100
	Acurácia	1,0000	0,0909	0,9090	1,0000	0,9000	1,0000
Usb_Conn	TP	10	10	6	10	10	10
	TN	1**00**	1**00**	1**00**	9**0**	1**6**	1**00**
	Acurácia	1,0000	1,0000	0,9636	0,9090	0,2663	1,0000

Tabela 31 - Tempo de Processamento para o Modelo A

Tempo de Processamento	
Tempo médio por Componente:	0,3896 s
Tempo por *bracket*:	8,5712 s

5.1.1 Discussão sobre os resultados do modelo A

Analisando a Tabela 30 percebe-se que apenas 2 elementos (Conn_3 e Gnd_1), obtiveram acurácia abaixo de 80%. Este resultado pode induzir a acreditar que o modelo é eficaz na detecção de defeitos, além disso, a taxa de falso alarmes é baixa, máximo de 14% (excluindo os componentes com baixa acurácia), colaborando com essa linha de pensamento. Porém, ao analisar a capacidade do método de classificar componentes defeituosos, percebe-se que o modelo é ineficaz,

apresentando diversos componentes (Antena, Cable_1, Gnd_2, Marca_dAgua, Rubber_1 e Rubber_2) com TP próximo ou igual 0. Assim, ao utilizar esse modelo na linha de produção diversos *brackets* defeituosos não seriam identificados.

No caso do componente Gnd_1, o mesmo já havia apresentado um IoU Score inferior aos demais componentes, mesmo com o watershed o seu valor foi de 0,7190 (Tabela 24), ou seja, a segmentação do componente não foi ideal, prejudicando toda a análise.

A Figura 50 mostra uma comparação entre um caso de componente defeituoso (ausência do Gnd_1) e um componente normal, além de ambas as figuras serem semelhantes, ainda existe a possibilidade de que um conector próximo influencie a análise. Mesmo com essas dificuldades, a técnica de subtração de imagem conseguiu uma acurácia próxima à 78%, deste modo, melhorando a segmentação do componente é possível conseguir um desempenho aceitável na linha de produção.

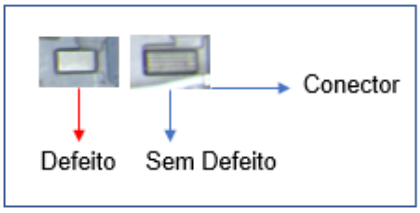


Figura 50 - Inspeção Gnd_1

Em relação ao componente Conn_3, a maioria dos casos de defeitos estão relacionados a um encaixe impreciso do conector. A segmentação semântica pode localizar o componente que está mal encaixado, estando o componente válido de acordo com o treinamento. A Figura 51 exemplifica o que foi escrito, percebe-se também que o componente defeituoso parece ser um pouco mais escuro, como no treino, o DA foi configurado para variar o brilho, esse parâmetro pode ter contribuído com os erros do modelo.

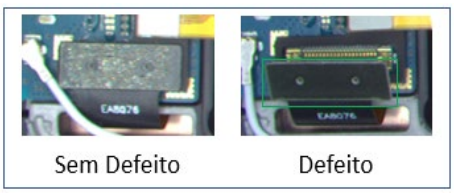


Figura 51 - Análise Conn_3

No caso de componentes como a Antena, o problema está na região de procura, pois para compensar o pequeno erro no registro, a região de procura é expandida. Existe uma outra antena muito próxima a antena em análise e em todas as imagens essa segunda antena não apresenta defeitos, deste modo o TP igual a zero é explicado. Analisar ambas as antenas seria uma solução para o caso. A Figura 52 exemplifica o que foi comentado no parágrafo.

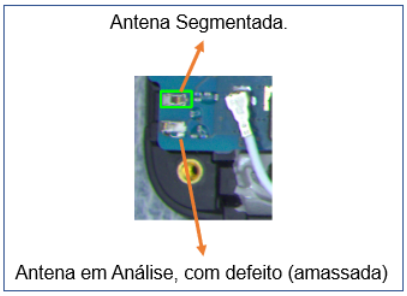


Figura 52 - Erro Análise Antena

O componente Cable_1 também apresentou baixo desempenho detectando erros, porém considerando o método baseado na subtração em relação ao *template*, a acurácia foi de praticamente 100%. Assim, esse método poderia ser utilizado para avaliar esse componente, ou poderia ser utilizado para criar uma base de treinamento robusta para os métodos baseados em CNN.

Um raciocínio semelhante pode ser considerado no caso dos componentes Rubber_1 e Marca_dAgua. Nestes componentes alguns métodos tiveram um desempenho interessante na análise de defeitos, a subtração em relação ao *template,* por exemplo, obteve 100% de acurácia e identificou 100% dos defeitos em ambos os componentes.

Para o componente Gnd_2 os métodos CAE e WGAN-GP tiveram uma acurácia próxima à 70%, com uma detecção razoável de defeitos. O Componente Gnd_2 também é um componente que, quando defeituoso, é semelhante a um caso de não defeito, conforme pode ser observado na Figura 53.

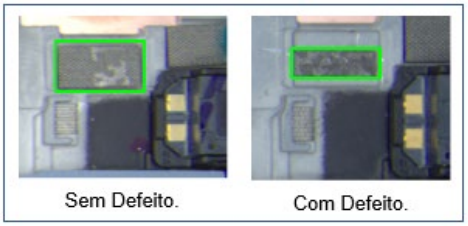


Figura 53 - Análise Gnd_2

Já no caso do componente Rubber_2, o método de subtração apresentou o melhor resultado, porém conseguiu encontrar apenas 50% dos defeitos. O *IoU Score* desse componente foi de 0,7063, prejudicando assim toda a análise, principalmente as análises dos métodos baseados em subtração. No caso deste componente, não se tentou utilizar nenhum processamento extra para melhorar a segmentação, como foi proposto em outros componentes.

No estudo de caso deste trabalho, na maioria dos componentes, conseguiu-se um bom desempenho utilizando-se apenas o método de subtração em relação ao *template* (Acurácia média de 95,70% e detecção de erros de 68,12%). Esse método, necessita que o *template* seja sempre atualizado, além disso, é susceptível a mudanças de cor dos componentes, marcas inseridas manualmente em outros processos de inspeção visual e pequenos desalinhamentos aceitáveis.

Embora existam essas dificuldades, o uso dessa técnica é interessante no início da produção de novos modelos, pois é fácil de parametrizar, sendo necessário apenas o treinamento em relação à U-Net. Outra aplicação interessante para o método é a sua capacidade de criar uma base de dados para o treinamento de métodos baseados em CNN, pode-se usar a U-Net e subtração para classificar automaticamente um conjunto grande de imagens e inspecionar manualmente, removendo os erros que existirem.

A proposta do uso do comitê, não melhorou o desempenho em nenhum dos componentes estudados, possivelmente devido ao número reduzido de imagens utilizadas nos treinamentos dos métodos baseados em CNN. A melhor escolha para aumentar o desempenho seria o uso de apenas um método para cada componente. Assim, o software deveria ser configurável, associando o melhor método para o determinado componente. Por exemplo, para o componente Gnd_3 o CAE retornou apenas 4 falsos alarmes e classificou corretamente todos os defeitos, enquanto que o comitê obteve um resultado inferior.

Conforme discutido, o método baseado na subtração em relação ao *template*, apresentou os melhores resultados. Esse algoritmo, necessita que o componente seja segmentado de maneira precisa, a junção U-Net e *watershed* não conseguiu um resultado eficiente para todos os casos estudados. Muitos trabalhos estão utilizando o modelo CNN YOLO (*You Only Look Once*), onde a detecção e classificação dos componentes é feita através de modificações nesta CNN.

Analisando a Tabela 32, com alguns exemplos de trabalhos citados na revisão bibliográfica, percebe-se que é necessário um número bem maior de imagens para o treinamento do modelo quando comparado ao usado pela U-Net. Em [89] são apresentados estudos de casos com diversos tamanhos de base de dados, pode-se perceber que o desempenho do YOLO melhora de acordo com a disponibilidade de dados para o treinamento.

Pode-se pensar em usar o DA para aumentar a base de dados, porém a configuração da DA utilizada nesse trabalho (rotação, translação, configuração do brilho, etc), segundo [90] é insuficiente para trazer ganhos à habilidade de generalização das CNN. Mesmo o uso do TF (*Transfer Learning*) pode ser desaconselhável, pois a análise de PCB de *smartphones* representa um domínio específico quando comparado aos domínios que possuem o YOLO pré-treinado.

Baseado nos números da Tabela 32 e na descrição dos últimos parágrafos, é inviável treinar um YOLO para segmentar os componentes do *bracket*, pelo menos no lançamento de um novo produto (NPI – *New Product Introduction*) [91]. Mesmo considerando produtos, cujos processos produtivos estão maduros, anotar 22 componentes, em por exemplo 510 imagens, é extremamente demorado e tedioso. Baseado nesses argumentos, reforça-se a importância do método apresentado nesse trabalho, onde foi possível segmentar e classificar os componentes com poucas imagens de treinamento, podendo inclusive ser utilizado na geração de um banco de imagens apropriado para técnicas mais eficientes.

Tabela 32 - Exemplo de trabalhos com YOLO.

Autor / Ano de publicação	Quantidade de Imagens	Resultado
Li et al. / 2019 [76]	834	mAP: 96,73%.
Adibhatla et al. / 2020 [62]	11000	Acurácia: 98,79%.
Khare et al. / 2020 [73]	510	Acurácia: 75,48 %.
Xie et al. / 2021 [77]	693	mAP: 99,71%.

Neste trabalho utilizou-se métodos de CNN com arquiteturas propostas em outros trabalhos, em nenhum momento tentou-se modificar as arquiteturas com inclusão ou exclusão de novas camadas convolucionais. Segundo [90], quando tenta-se um aprendizado com poucas amostras (FSL – *Few Samples Learning*), sem que exista um projeto específico da CNN, o modelo irá apresentar *overfitting*. Deste modo, o desempenho ruim das técnicas baseadas em CNN deste trabalho (CAE, VGG *Cluster* e WGAN-GP) podem ser explicadas.

Além de não alterar a arquitetura, é proposto pelo presente trabalho, o uso de poucos parâmetros para a configuração do modelo. Deste modo, os parâmetros padrões da *API Keras* foram utilizados, a configuração dos hiper parâmetros são essenciais no desempenho dos modelos CNN [15], ou seja, a falta de parametrização também pode ter interferido nos modelos utilizados.

5.2 Modelo B

As imagens utilizadas para a avaliação do modelo de *bracket* B não possuem nenhuma anomalia. Então, as mesmas serão utilizadas somente para avaliar o resultado do registro do *bracket* e segmentação dos componentes. Nesse caso, existem mais imagens para a avaliação, não sendo utilizado a DA para gerar o conjunto de teste. A Tabela 33 apresenta os resultados do estudo.

Tabela 33 - Resultado do IoU Score para modelo B

Modelo B				
Total de imagens		54		
Imagens utilizadas no treinamento/validação		12		
Imagens utilizadas no teste		42		
Dados do DA (*Data Augmentation*)				
Imagens Originais	**Tamanho do *Batch***	**Rotação (Graus)**	**Translação Vertical (%)**	**Translação Horizontal (%)**
12	4	5	0,02	0,02
Dados do treino da U-Net				
Épocas	***Learn Rate***	**Otimizador**	***Loss***	***Métricas***
200	0,001	Adam	*Binary Crossentropy*	*Accuracy*
Resultados do Treinamento				
Componente		**Acurácia**		***IoU Score***
bracket		0,9719		0,9825 ±0,0032
Motor		0,9910		0,8192 ±0,1523
Borracha		0,9933		0,8688 ±0,0739
Conector de Áudio		0,9931		0,8688 ±0,0473
Parafuso		0,9642		0,5867 ±0,2731

5.2.1 Discussão dos Resultado do Modelo B

Para o treinamento do Modelo B também se utilizou poucas imagens para o treinamento e validação, novamente o processo de DA foi responsável por aumentar esses conjuntos durante o treinamento, seguindo as mesmas configurações do Modelo A. Assim, deixando mais imagens para o teste, pretende-se comprovar a eficácia do método de registro, mesmo quando aplicado em imagens originais.

Excluindo-se o parafuso, todos os outros componentes tiveram um *IoU Score* superior a 0,8000. O baixo desempenho do método para o componente Parafuso pode ser explicado pela falta de iluminação nesse componente. Provavelmente, o

sensor de gatilho da foto precisa ser ajustado para o Modelo B, disparando quando todo o *bracket* estiver uniformemente iluminado.

Como os resultados foram semelhantes aos apresentados no modelo A, o método utilizado pode ser aplicado em outros modelos de *smartphone*, respeitando apenas a configuração de hardware. Assim, apenas algumas imagens são necessárias para treinar o método e o mesmo pode ser aplicado desde o início da produção do novo modelo.

6 CONCLUSÕES

Nos últimos anos a indústria eletrônica tem vivenciado um crescimento no aparecimento de novos produtos e novos fabricantes. Os novos produtos trazem consigo um aumento na complexidade de produção dos mesmos, enquanto que novos fabricantes acarretam uma maior competição por uma fatia de mercado.

Deste modo, a garantia da qualidade dos produtos torna-se um fator fundamental na sobrevivência da indústria eletrônica. A garantia de qualidade diminui o custo de produção, sendo um fator fundamental na confiança da marca adquirida por parte dos consumidores.

A qualidade de um produto eletrônico está intimamente relacionada à qualidade da sua PCB (*Printed Circuit Board)*. Atualmente, as indústrias eletrônicas adotam basicamente, dois tipos de técnicas para avaliar a integridade das PCB: inspeção por ICT (*In Circuit Test*) e inspeção visual.

A inspeção por ICT não garante a cobertura de todo o circuito da PCB. Já a inspeção visual realizada pelas máquinas de AOI, requerem profissionais bem treinados. Por fim, a inspeção visual realizada por operadores possui um caráter subjetivo, exigindo conhecimento do profissional e alta concentração em todo o turno de trabalho.

Neste contexto, o presente trabalho propôs um método híbrido que utiliza métodos tradicionais e métodos baseadas em CNN (*Convolutional Neural Networks*), capazes de identificar anomalias em PCB. Embora o comitê possa ser aplicado em qualquer produto eletrônicos, os resultados foram obtidos avaliando-se a PCB de *smartphones*, cujas a imagens foram retiradas em um ponto específico do processo produtivo.

As arquiteturas CNN exigem grandes quantidades de dados para o treinamento, considerando a dificuldade de se encontrar, em quantidades suficientes, imagens de componentes defeituosos na indústria eletrônica, foram considerados no trabalho métodos capazes de serem treinados somente com imagens não defeituosas. Além disso, graças ao constante lançamento de novos produtos, foram utilizadas técnicas para o treinamento com poucas imagens, tais como DA (*Data Augmentation*) e TL (*Transfer Learning*).

Considerando os 22 componentes avaliados, conseguiu-se uma acurácia média de 90,87% e uma taxa de falsos alarmes de 5,31%. O tempo de treinamento do comitê foi de aproximadamente 20 h, enquanto que o tempo de predição foi de 8,57 s. Embora a acurácia seja considerável, o modelo não conseguiu uma eficácia significativa na detecção de defeitos em ao menos 6 componentes. Sendo que nesses casos, a diferença entre um componente defeituoso e um componente sem anomalias é muito pequena ou observou-se uma dificuldade na segmentação do componente, apresentando *IoU Score* próximo a 0,7000. Por fim, a junção das técnicas no comitê apresentou um desempenho inferior ao uso de uma técnica específica para cada componente.

Embora o resultado do comitê tenha apresentado pouca eficácia na detecção de anomalias em relação a alguns componentes, o modelo pode ser utilizado no início do processo produtivo, uma vez que pode ser treinado com poucas imagens e em pouco tempo. Além disso, o modelo pode ser utilizado para segmentar componentes não defeituosos, de modo a criar uma base de dados para modelos CNN mais avançados, evitando o trabalho manual de marcação dos componentes.

Em trabalhos futuros pretende-se modificar as arquiteturas das CNN utilizadas, com a adição de novas camadas convolucionais espera-se que os modelos apresentem melhores resultados. Além disso, será estudada a parametrização do treinamento de acordo com a problema a ser resolvido, propondo funções de custo e parâmetros mais eficientes do os padrões definidos pela *API Keras*. Por fim, deve-se propor métodos mais eficientes de DA e produção sintéticas de imagens com componentes defeituosos enriquecendo a base de dados de treinamento.

7 REFERÊNCIAS

[1] J. Li, J. Gu, Z. Huang, e J. Wen, “Application Research of Improved YOLO V3 Algorithm in PCB Electronic Component Detection”, *Appl. Sci.*, vol. 9, nº 18, Art. nº 18, jan. 2019, doi: 10.3390/app9183750.

[2] D. B. Anitha e M. Rao, “A survey on defect detection in bare PCB and assembled PCB using image processing techniques”, in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, mar. 2017, p. 39–43. doi: 10.1109/WiSPNET.2017.8299715.

[3] E. H. Yuk, S. H. Park, C.-S. Park, e J.-G. Baek, “Feature-Learning-Based Printed Circuit Board Inspection via Speeded-Up Robust Features and Random Forest”, *Appl. Sci.*, vol. 8, nº 6, Art. nº 6, jun. 2018, doi: 10.3390/app8060932.

[4] A.-A. I. M. Hassanin, F. E. Abd El-Samie, e G. M. El Banby, “A real-time approach for automatic defect detection from PCBs based on SURF features and morphological operations”, *Multimed. Tools Appl.*, vol. 78, nº 24, p. 34437–34457, dez. 2019, doi: 10.1007/s11042-019-08097-9.

[5] R. A. Martins, *Conceitos Básicos de Controle Estatístico da Qualidade*, 1ª edição. Edufscar, 2010.

[6] M. Ke, C. Lin, e Q. Huang, “Anomaly detection of Logo images in the mobile phone using convolutional autoencoder”, in *2017 4th International Conference on Systems and Informatics (ICSAI)*, nov. 2017, p. 1163–1168. doi: 10.1109/ICSAI.2017.8248461.

[7] D. Li, C. Li, C. Chen, e Z. Zhao, “Semantic Segmentation of a Printed Circuit Board for Component Recognition Based on Depth Images”, *Sensors*, vol. 20, nº 18, Art. nº 18, jan. 2020, doi: 10.3390/s20185318.

[8] A. R. de Mello e M. R. Stemmer, “Inspecting surface mounted devices using k nearest neighbor and Multilayer Perceptron”, in *2015 IEEE 24th International Symposium on Industrial Electronics (ISIE)*, jun. 2015, p. 950–955. doi: 10.1109/ISIE.2015.7281599.

[9] J. Richter, D. Streitferdt, e E. Rozova, “On the development of intelligent optical inspections”, in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, jan. 2017, p. 1–6. doi: 10.1109/CCWC.2017.7868455.

[10] V. Chaudhary, I. R. Dave, e K. P. Upla, “Automatic visual inspection of printed circuit board for defect detection and classification”, in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, mar. 2017, p. 732–737. doi: 10.1109/WiSPNET.2017.8299858.

[11] W. Shi, L. Zhang, Y. Li, e H. Liu, “Adversarial semi-supervised learning method for printed circuit board unknown defect detection”, *J. Eng.*, vol. 2020, nº 13, p. 505–510, 2020, doi: 10.1049/joe.2019.1181.

[12] F. P. da S. Costa, “Projeto e reconfiguração de linhas de montagem para novos produtos usando princípios Lean Thinking numa empresa de componentes eletrónicos”, 2019, Acessado: 8 de dezembro de 2020. [Online]. Disponível em: http://repositorium.sdum.uminho.pt/

[13] N. O’Mahony *et al.*, “Deep Learning vs. Traditional Computer Vision”, in *Advances in Computer Vision*, Cham, 2020, p. 128–144. doi: 10.1007/978-3-030-17795-9_10.

[14] S. Skansi, *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*, 1st ed. 2018 edition. New York, NY: Springer, 2018.

[15] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*, Softcover reprint of the original 1st ed. 2018 edição. New York: Springer, 2019.
[16] B. AG, “Global Shutter, Rolling Shutter | Basler”, *Basler AG*. /en/sales-support/downloads/document-downloads/global-shutter-rolling-shutter/ (acessado 25 de outubro de 2020).
[17] “Global Shutter Imaging | Teledyne DALSA”. https://www.teledynedalsa.com/en/learn/knowledge-center/global-shutter-imaging/ (acessado 25 de outubro de 2020).
[18] T. Le, N. Le, e Y. M. Jang, “Performance of rolling shutter and global shutter camera in optical camera communications”, in *2015 International Conference on Information and Communication Technology Convergence (ICTC)*, out. 2015, p. 124–128. doi: 10.1109/ICTC.2015.7354509.
[19] R. C. Gonzalez e R. E. Woods, *Processamento Digital de Imagens*, 3ª edição. Pearson Universidades, 2009.
[20] R. Klette, *Concise Computer Vision: An Introduction Into Theory and Algorithms*. London: Springer, 2014.
[21] D. Tsai e C. Huang, “Defect Detection in Electronic Surfaces Using Template-Based Fourier Image Reconstruction”, *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 9, nº 1, p. 163–172, jan. 2019, doi: 10.1109/TCPMT.2018.2873744.
[22] L. N. de Castro e D. G. Ferrari, *Introdução à mineração de dados*, 1º ed. São Paulo: Editora Saraiva, 2016.
[23] Isabel, “‘Nobel da Computação’ vai para os pais do Deep Learning”, *Instituto de Engenharia*, 1º de abril de 2019. https://www.institutodeengenharia.org.br/site/2019/04/01/nobel-da-computacao-vai-para-os-pais-do-deep-learning/ (acessado 12 de dezembro de 2020).
[24] “Deep Learning Book”, *Deep Learning Book*. http://deeplearningbook.com.br/ (acessado 24 de dezembro de 2020).
[25] S. Albawi, T. A. Mohammed, e S. Al-Zawi, “Understanding of a convolutional neural network”, in *2017 International Conference on Engineering and Technology (ICET)*, ago. 2017, p. 1–6. doi: 10.1109/ICEngTechnol.2017.8308186.
[26] J. Koushik, “Understanding Convolutional Neural Networks”, maio 2016, Acessado: 22 de dezembro de 2020. [Online]. Disponível em: https://arxiv.org/abs/1605.09081v1
[27] V. Dumoulin e F. Visin, “A guide to convolution arithmetic for deep learning”, *ArXiv160307285 Cs Stat*, jan. 2018, Acessado: 21 de fevereiro de 2022. [Online]. Disponível em: http://arxiv.org/abs/1603.07285
[28] R. Yamashita, M. Nishio, R. K. G. Do, e K. Togashi, “Convolutional neural networks: an overview and application in radiology”, *Insights Imaging*, vol. 9, nº 4, Art. nº 4, ago. 2018, doi: 10.1007/s13244-018-0639-9.
[29] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way”, *Medium*, 17 de dezembro de 2018. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (acessado 18 de janeiro de 2021).
[30] Y. Ho e S. Wookey, “The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling”, *IEEE Access*, vol. PP, p. 1–1, dez. 2019, doi: 10.1109/ACCESS.2019.2962617.
[31] N. Gowdra, R. Sinha, S. MacDonell, e W. Yan, “Maximum Categorical Cross Entropy (MCCE): A noise-robust alternative loss function to mitigate racial bias in Convolutional Neural Networks (CNNs) by reducing overfitting”, set. 2020,

Acessado: 24 de dezembro de 2021. [Online]. Disponível em: https://openreview.net/forum?id=1IBgFQbj7y

[32] D. P. Kingma e J. Ba, “Adam: A Method for Stochastic Optimization”, *ArXiv14126980 Cs*, jan. 2017, Acessado: 24 de dezembro de 2021. [Online]. Disponível em: http://arxiv.org/abs/1412.6980

[33] “Complete Guide to Transposed Convolutions in CNN Models”, *Analytics India Magazine*, 5 de setembro de 2021. https://analyticsindiamag.com/complete-guide-to-transposed-convolutions-in-cnn-models/ (acessado 21 de fevereiro de 2022).

[34] Z. Chen, C. K. Yeo, B. S. Lee, e C. T. Lau, “Autoencoder-based network anomaly detection”, in *2018 Wireless Telecommunications Symposium (WTS)*, abr. 2018, p. 1–5. doi: 10.1109/WTS.2018.8363930.

[35] I. J. Goodfellow *et al.*, “Generative Adversarial Networks”, *ArXiv14062661 Cs Stat*, jun. 2014, Acessado: 20 de janeiro de 2021. [Online]. Disponível em: http://arxiv.org/abs/1406.2661

[36] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, e X. Chen, “Improved Techniques for Training GANs”, *ArXiv160603498 Cs*, jun. 2016, Acessado: 20 de janeiro de 2021. [Online]. Disponível em: http://arxiv.org/abs/1606.03498

[37] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, e V. R. Chandrasekhar, “Efficient GAN-Based Anomaly Detection”, *ArXiv180206222 Cs Stat*, maio 2019, Acessado: 20 de janeiro de 2021. [Online]. Disponível em: http://arxiv.org/abs/1802.06222

[38] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, e U. Schmidt-Erfurth, “f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks”, *Med. Image Anal.*, vol. 54, p. 30–44, maio 2019, doi: 10.1016/j.media.2019.01.010.

[39] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, e A. Courville, “Improved Training of Wasserstein GANs”, *ArXiv170400028 Cs Stat*, dez. 2017, Acessado: 27 de dezembro de 2021. [Online]. Disponível em: http://arxiv.org/abs/1704.00028

[40] D. Foster, *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play*, 1st edition. Sebastopol, CA: O’Reilly Media, 2019.

[41] M. Arjovsky, S. Chintala, e L. Bottou, “Wasserstein GAN”, *ArXiv170107875 Cs Stat*, dez. 2017, Acessado: 27 de dezembro de 2021. [Online]. Disponível em: http://arxiv.org/abs/1701.07875

[42] P. Iakubovskii, *qubvel/segmentation_models*. 2021. Acessado: 21 de dezembro de 2021. [Online]. Disponível em: https://github.com/qubvel/segmentation_models

[43] T. Liang *et al.*, “CBNetV2: A Composite Backbone Network Architecture for Object Detection”, *ArXiv210700420 Cs*, jul. 2021, Acessado: 21 de dezembro de 2021. [Online]. Disponível em: http://arxiv.org/abs/2107.00420

[44] A. Benali Amjoud e M. Amrouch, “Convolutional Neural Networks Backbones for Object Detection”, in *Image and Signal Processing*, Cham, 2020, p. 282–289. doi: 10.1007/978-3-030-51935-3_30.

[45] N. Tajbakhsh *et al.*, “Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?”, *IEEE Trans. Med. Imaging*, vol. 35, nº 5, p. 1299–1312, maio 2016, doi: 10.1109/TMI.2016.2535302.

[46] “OpenCV: Understanding Features”. https://docs.opencv.org/3.4/df/d54/tutorial_py_features_meaning.html (acessado 27 de outubro de 2020).
[47] “OpenCV: Introduction to SIFT (Scale-Invariant Feature Transform)”. https://docs.opencv.org/3.4/da/df5/tutorial_py_sift_intro.html (acessado 27 de outubro de 2020).
[48] O. Ronneberger, P. Fischer, e T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation”, maio 2015, Acessado: 23 de novembro de 2020. [Online]. Disponível em: https://arxiv.org/abs/1505.04597v1
[49] K. Parvati, B. S. Prakasa Rao, e M. Mariya Das, “Image Segmentation Using Gray-Scale Morphology and Marker-Controlled Watershed Transformation”, *Discrete Dyn. Nat. Soc.*, vol. 2008, p. e384346, jan. 2009, doi: 10.1155/2008/384346.
[50] “OpenCV: Image Segmentation with Watershed Algorithm”. https://docs.opencv.org/4.4.0/d3/db4/tutorial_py_watershed.html (acessado 15 de dezembro de 2021).
[51] J. Redmon, S. Divvala, R. Girshick, e A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”, 2016, p. 779–788. Acessado: 29 de novembro de 2020. [Online]. Disponível em: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html
[52] “3.3. Metrics and scoring: quantifying the quality of predictions”, *scikit-learn*. https://scikit-learn/stable/modules/model_evaluation.html (acessado 28 de dezembro de 2021).
[53] L. Dai, Q. Guan, e H. Liu, “Robust image registration of printed circuit boards using improved SIFT-PSO algorithm”, *J. Eng.*, vol. 2018, nº 16, p. 1793–1797, 2018, doi: 10.1049/joe.2018.8274.
[54] G. Hua, W. Huang, e H. Liu, “Accurate image registration method for PCB defects detection”, *J. Eng.*, vol. 2018, nº 16, p. 1662–1667, 2018, doi: 10.1049/joe.2018.8272.
[55] J. Zhu, A. Wu, e X. Liu, “Printed circuit board defect visual detection based on wavelet denoising”, *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 392, p. 062055, ago. 2018, doi: 10.1088/1757-899X/392/6/062055.
[56] D. Cho, *Image Denoising Using Wavelet Transforms*. Saarbrücken: VDM Verlag Dr. Mueller E.K., 2008.
[57] M. H. Annaby, Y. M. Fouda, e M. A. Rushdi, “Improved Normalized Cross-Correlation for Defect Detection in Printed-Circuit Boards”, *IEEE Trans. Semicond. Manuf.*, vol. 32, nº 2, p. 199–211, maio 2019, doi: 10.1109/TSM.2019.2911062.
[58] K. R. Rao e P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press, 2014.
[59] Q. He, “Defect detection of PCB based on Bayes feature fusion”, *J. Eng.*, vol. 2018, ago. 2018, doi: 10.1049/joe.2018.8270.
[60] T. J. M. D. Oliveira, M. A. Wehrmeister, e B. T. Nassu, “Detecting Modifications in Printed Circuit Boards from Fuel Pump Controllers”, in *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, out. 2017, p. 87–94. doi: 10.1109/SIBGRAPI.2017.18.
[61] D. Li, C. Li, C. Chen, e Z. Zhao, “Semantic Segmentation of a Printed Circuit Board for Component Recognition Based on Depth Images”, *Sensors*, vol. 20, nº 18, Art. nº 18, jan. 2020, doi: 10.3390/s20185318.

[62] V. A. Adibhatla, H.-C. Chih, C.-C. Hsu, J. Cheng, M. F. Abbod, e J.-S. Shieh, “Defect Detection in Printed Circuit Boards Using You-Only-Look-Once Convolutional Neural Networks”, *Electronics*, vol. 9, nº 9, Art. nº 9, set. 2020, doi: 10.3390/electronics9091547.

[63] I. Volkau, A. Mujeeb, D. Wenting, E. Marius, e S. Alexei, “Detection Defect in Printed Circuit Boards using Unsupervised Feature Extraction Upon Transfer Learning”, in *2019 International Conference on Cyberworlds (CW)*, out. 2019, p. 101–108. doi: 10.1109/CW.2019.00025.

[64] M. A. Mallaiyan Sathiaseelan, O. P. Paradis, S. Taheri, e N. Asadizanjani, “Why Is Deep Learning Challenging for Printed Circuit Board (PCB) Component Recognition and How Can We Address It?”, *Cryptography*, vol. 5, nº 1, Art. nº 1, mar. 2021, doi: 10.3390/cryptography5010009.

[65] H. Wang, M. Li, F. Ma, S.-L. Huang, e L. Zhang, “Unsupervised anomaly detection via generative adversarial networks: poster abstract”, in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, New York, NY, USA, abr. 2019, p. 313–314. doi: 10.1145/3302506.3312605.

[66] C.-W. Kuo, J. Ashmore, D. Huggins, e Z. Kira, “Data-Efficient Graph Embedding Learning for PCB Component Detection”, apresentado em WACV19, nov. 2018. Acessado: 14 de abril de 2021. [Online]. Disponível em: https://arxiv.org/abs/1811.06994v2

[67] L. Zhang *et al.*, “Convolutional neural network-based multi-label classification of PCB defects”, *J. Eng.*, vol. 2018, nº 16, p. 1612–1616, 2018, doi: 10.1049/joe.2018.8279.

[68] D. Lim, Y. Kim, e T. Park, “SMD Classification for Automated Optical Inspection Machine Using Convolution Neural Network”, in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, fev. 2019, p. 395–398. doi: 10.1109/IRC.2019.00072.

[69] W. Shi, Z. Lu, W. Wu, e H. Liu, “Single-shot detector with enriched semantics for PCB tiny defect detection”, *J. Eng.*, vol. 2020, nº 13, p. 366–372, 2020, doi: 10.1049/joe.2019.1180.

[70] S. Khalilian, Y. Hallaj, A. Balouchestani, H. Karshenas, e A. Mohammadi, “PCB Defect Detection Using Denoising Convolutional Autoencoders”, in *2020 International Conference on Machine Vision and Image Processing (MVIP)*, fev. 2020, p. 1–5. doi: 10.1109/MVIP49855.2020.9187485.

[71] B. Xia, J. Cao, e C. Wang, “SSIM-NET: Real-Time PCB Defect Detection Based on SSIM and MobileNet-V3”, in *2019 2nd World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM)*, nov. 2019, p. 756–759. doi: 10.1109/WCMEIM48965.2019.00159.

[72] D. Brunet, E. R. Vrscay, e Zhou Wang, “On the Mathematical Properties of the Structural Similarity Index”, *IEEE Trans. Image Process.*, vol. 21, nº 4, p. 1488–1499, abr. 2012, doi: 10.1109/TIP.2011.2173206.

[73] T. Khare, V. Bahel, e A. C. Phadke, “PCB-Fire: Automated Classification and Fault Detection in PCB”, in *2020 Third International Conference on Multimedia Processing, Communication Information Technology (MPCIT)*, dez. 2020, p. 123–128. doi: 10.1109/MPCIT51588.2020.9350324.

[74] X. Wu, Y. Ge, Q. Zhang, e D. Zhang, “PCB Defect Detection Using Deep Learning Methods”, in *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, maio 2021, p. 873–876. doi: 10.1109/CSCWD49262.2021.9437846.

[75] L. H. de S. Silva, G. O. de A. Azevedo, B. J. T. Fernandes, B. L. D. Bezerra, E. B. Lima, e S. C. Oliveira, “Automatic Optical Inspection for Defective PCB Detection Using Transfer Learning”, in *2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, nov. 2019, p. 1–6. doi: 10.1109/LA-CCI47412.2019.9037036.

[76] Y.-T. Li, P. Kuo, e J.-I. Guo, “Automatic Industry PCB Board DIP Process Defect Detection with Deep Ensemble Method”, in *2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)*, jun. 2020, p. 453–459. doi: 10.1109/ISIE45063.2020.9152533.

[77] H. Xie, Y. Li, X. Li, e L. He, “A Method for Surface Defect Detection of Printed Circuit Board Based on Improved YOLOv4”, in *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, mar. 2021, p. 851–857. doi: 10.1109/ICBAIE52039.2021.9390006.

[78] “Calculating Camera Sensor Resolution and Lens Focal Length”. https://www.ni.com/pt-br/support/documentation/supplemental/18/calculating-camera-sensor-resolution-and-lens-focal-length.html (acessado 13 de fevereiro de 2022).

[79] “Sensors and Lenses | Edmund Optics”. https://www.edmundoptics.com/knowledge-center/application-notes/imaging/sensors-and-lenses/ (acessado 13 de fevereiro de 2022).

[80] C. Pramerdorfer e M. Kampel, “A dataset for computer-vision-based PCB analysis”, in *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*, maio 2015, p. 378–381. doi: 10.1109/MVA.2015.7153209.

[81] S. Tang, F. He, X. Huang, e J. Yang, “Online PCB Defect Detector On A New PCB Defect Dataset”, *ArXiv190206197 Cs*, fev. 2019, Acessado: 29 de setembro de 2020. [Online]. Disponível em: http://arxiv.org/abs/1902.06197

[82] “OpenCV: Camera Calibration and 3D Reconstruction”. https://docs.opencv.org/4.4.0/d9/d0c/group__calib3d.html#ga4abc2ece9fab9398f2e560d53c8c9780 (acessado 22 de novembro de 2020).

[83] “OpenCV: Operations on arrays”. https://docs.opencv.org/4.4.0/d2/de8/group__core__array.html#gad327659ac03e5fd6894b90025e6900a7 (acessado 22 de novembro de 2020).

[84] “OpenCV: cv::BFMatcher Class Reference”. https://docs.opencv.org/4.4.0/d3/da1/classcv_1_1BFMatcher.html (acessado 8 de dezembro de 2021).

[85] “OpenCV: Geometric Image Transformations”. https://docs.opencv.org/4.4.0/da/d54/group__imgproc__transform.html#ga20f62aa3235d869c9956436c870893ae (acessado 8 de dezembro de 2021).

[86] “tf.keras.preprocessing.image.ImageDataGenerator | TensorFlow Core v2.7.0”, *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator?hl=pt-br (acessado 8 de dezembro de 2021).

[87] “ImageNet”. https://www.image-net.org/ (acessado 22 de dezembro de 2021).

[88] D.-M. Tsai e P.-H. Jen, “Autoencoder-based anomaly detection for surface defect inspection”, *Adv. Eng. Inform.*, vol. 48, p. 101272, abr. 2021, doi: 10.1016/j.aei.2021.101272.

[89] S. D. Kulik e A. N. Shtanko, “Experiments with Neural Net Object Detection System YOLO on Small Training Datasets for Intelligent Robotics”, in *Advanced Technologies in Robotics and Intelligent Systems*, Cham, 2020, p. 157–162. doi: 10.1007/978-3-030-33491-8_19.

[90] J. Lu, P. Gong, J. Ye, e C. Zhang, “Learning from Very Few Samples: A Survey”, set. 2020, Acessado: 11 de março de 2021. [Online]. Disponível em: https://arxiv.org/abs/2009.02653v2

[91] B. Mroz e R. Mroz, *New Product Introduction - The Complete Task List*. Customer Manufacturing Group, 2011.