University of Stavanger

SHAM YOHANNES BEEMNET

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Defect Detection in Industrial Images

Masters - Computer Science - August 2023

I, **Sham Yohannes Beemnet**, declare that this thesis titled, “Defect Detection in Industrial Images” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a master’s degree at the University of Stavanger.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

"You can have data without information, but you cannot have information without data."

– Daniel Keys Moran

Abstract

This thesis aims to identify defects in industrial images using Printed Circuit Board(PCB) images. Defect detection in industrial images is crucial in this era that we are living in today. There is a rising demand for automated ways to detect and classify defects in the place of labor that are becoming less available and more expensive. Additionally, this will enhance the output and raise overall product quality.

This thesis presents a method for detecting defects in PCB images utilizing autoencoders, variational autoencoders, convolutional neural networks, and classifiers like K-Nearest Neighbours and Random forests. The study's primary objective is to feed the classifiers with data from the autoencoder's latent space before passing it to the decoder to reconstruct the image and determine the results. Different models with varied hyperparameters and architectures train the different autoencoders.

The study's findings indicate that The VAE with convolutional neural network performed best from the four methods used in this thesis with 98.5% accuracy. This result means that using the latent space before the data gets reconstructed in the decoder to train a model and classify defects proved better than using autoencoders in the conventional sense in the case of the Convolutional Neural network.

Acknowledgements

The journey of writing this thesis was made more accessible because of everyone around me. But a special thanks goes to my supervisor Arian Baloochestani, who has been available from the beginning to the end of my thesis; I am immensely grateful for all of the guidance, supervision availability, support, rapid feedback, and understanding I received from you, Arian; you are a fantastic person to work with. My special thanks also goes to my lovely wife and three wonderful kids for their inspiration and encouragement. I'd also like to thank all of the staff and instructors at the University of Stavanger for their hard work, as well as all of my co-students, students I met through group projects, and excellent friends I gained at this university.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

Defect detection has been used in all societies early on, whether in agriculture, construction materials, textiles, medicine, and so many other fields or disciplines in this world, to identify the anomalies within the collection of different kinds of products. People have always spent resources, time, and energy detecting defects. As the product in all sectors, as mentioned earlier, increases with the rise in the industry, increased complexity of products, and the increase in population, It gets more tedious and time-consuming to do the job manually. Moreover, defect detection calls for safe ways to detect anomalies or defects in some sectors, like medicine, as this can mean life and death if it fails to handle competently or efficiently. We have to deal with it properly and effectively, and we also require high-speed solutions to accommodate the world's fast-growing population. Furthermore, the need for a more effective and technological method that can significantly reduce the use of resources, time, and energy comes here. As a result, researchers have investigated and researched machine learning strategies to develop viable techniques of discovering industrial defect identification in the hope of replacing declining human labor and lowering associated labor costs. This led to the developments we are enjoying so far. As there are so many machine learning and deep learning techniques under development nowadays all over the globe. And with the current rate of advancement, we will undoubtedly see unthinkable progress in this area in the future. This shows the huge importance of the field in the time we are living in. Especially the use of Autoencoders as a tool to enhance

this development.

The potential that lies within the use of autoencoders in discovering defects in image datasets can be a way to break through and go far beyond the present achievement in this field, especially the idea that was motivational in this area is, instead of using the traditional way where the autoencoder reconstructs the original input image and then checks the reconstruction error to decide whether the image is defective or nondefective, It made more sense to choose to interrupt in the middle and try to examine if It can use the data in the compressed latent space before getting reconstructed and see what the outcome would be by feeding it to the different classifiers involved here. Therefore, This thesis was started with the excitement to see what would happen by implementing the method mentioned earlier.

1.2 Objectives

This thesis aims to research the use of autoencoders in defect identification. The following are the goals of this work:

- To comprehend and explain the design, operation, and importance of autoencoders for defect detection.
- Using a collection of PCB images, both with and without defects, to create and train an autoencoder model.
- To assess how well the trained model correctly recognizes defects in various test images.
- To assess the autoencoder-based methodology's accuracy and effectiveness compared to conventional defect detection techniques.
- To examine the existing model's implications, constraints, and possibilities for improvement, offering guidance for further research in this area.
- To extract the compressed latent space plot it and see the distribution of the data.
- To feed this latent space before it gets reconstructed in the autoencoders decoder, to the classifiers Convolutional neural net models and other classifiers like K-Nearest Neighbors and Randoms forests and see the result.

1.3 Outline

Below is given an outline of the organization and substance of this thesis. The first chapter, introduces the historical background, the motivation for defect detection, and how it was always there; with the rise of technology, it has advanced, and we expect it to improve even more. This chapter also touches on the importance of defect detection back in history and in this time we are living in and defines the goals of this thesis.

The second chapter, focuses mainly on the background and surveys of the methods used in this thesis. It briefly defines and illustrates all the theoretical approaches used one by one.

The third chapter, reviews some related work goes through the methods used, the challenges they faced and the results they achieved in the different articles and finally compares the accuracy found using those different methods.

The fourth chapter, Here the dataset used in this thesis is the main topic. Introduces The Printed Circuit Board (PCB) images and explains how they are preprocessed and used in this thesis.

The fifth chapter, the methodology section, provides information on the study's methodology, the thesis's design, and the numerous approaches utilized to detect defects in the image datasets.

The sixth chapter is the results and discussion chapter. The thesis results will be presented, evaluated, and compared to earlier works. The thesis' shortcomings are also discussed.

The seventh chapter is the conclusion part and here final thoughts are put to conclude the work done in this thesis and also suggests possible directions for future work

Chapter 2

Background

2.1 Introduction

2.2 Existing Approaches/Baselines

2.3 Basic concepts of Image analysis

The subject of image analysis involves using techniques to extract more complex data from images [Birchfield(2023)].

There are three main issues with image analysis. These are mainly Segmentation, Classification and Shape from X. We will briefly be introduced one by one as follows.

2.3.1 segmentation

Identifying the pixels that in an image correspond to each other, or which pixels are reflections of the same item in the scene, is a technique known as segmentation see figure 2.1 below. Lacking any model of the specific object in the picture that created the group of pixels, segmentation can also be seen as a bottom-up method whereby pixels are placed together based on low-level, local features of the pixels and those around them [Birchfield(2023)].

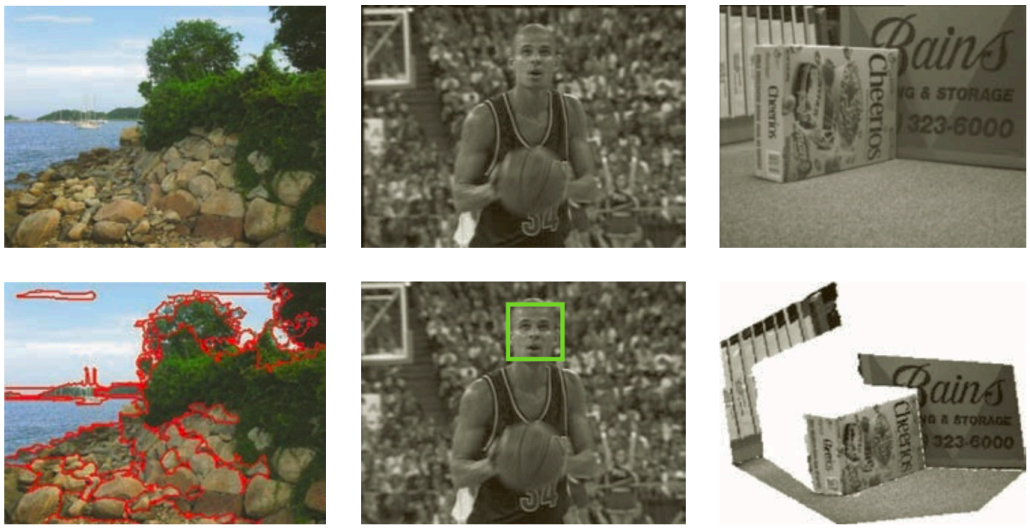

Figure 2.1: Segmentation, Classification and Shape from X [Birchfield(2023)].

2.3.2 Classification

As opposed to segmentation, the classification challenge entails identifying which pixels within an image are part of an earlier-created image. An individual trainer or other system is used to help create the model by which the pixels will ultimately be compared in the top-down classification process. If you've ever seen the outline of everyone's faces on a digital camera, that is the outcome of classification [Birchfield(2023)]. Image classification is particularly relevant to the thesis as images are classified as defective and non defective using the autoencoder and other forms of classification methods such as the K-Nearest Neighbors and Random forests. An example is shown in the figure 2.1

2.3.3 Shape From X

The third challenge, shape from X, see figure 2.1, attempts to reconstruct the scene's three-dimensional (3D) structure using any number of different methods (thus the "X"), including video, shading, stereo or texture. Image analysis uses ideas from linear algebra, statistics, geometry(projective), and the optimisation of functions to address these three categories of issues [Birchfield(2023)]

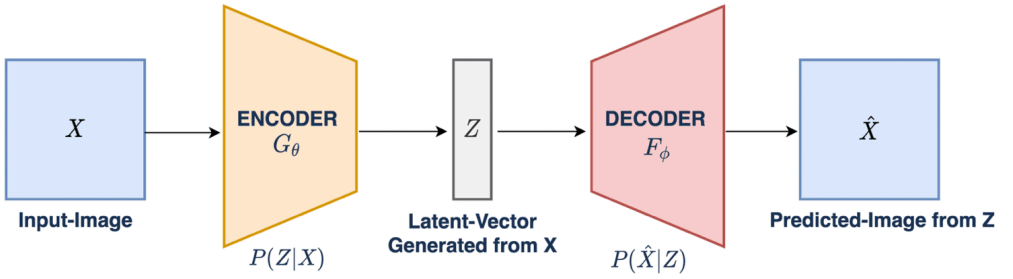

Figure 2.2: Autoencoder
[OpenCV(2023)].

2.4 Methods Used

In this part I will go over each of the models I've used in my thesis, why I chose to use them, how I put them into practice, and any notable changes I made if there are any.But first I let me try to introduce them one by one and give some illustrations to try to clearly explain them. The techniques I used include the following: Autencoders, Variational autoencoders, Denoising autoencoders, K-Nearest Neighbors (KNN), and Random Forests.

2.4.1 Autoencoders

Autoencoders are Neural network circuits which try to get an output out of the given input which in this case is the pcb images without much distortion.Autoencoders were introduced first in the 1980s by Hinton and the PDP group [Baldi(2012)]. It is a particular kind of neural network that is primarily made to encode the input into a compact and meaningful representation and then decode it so that the reconstructed input is as similar to the original as possible [Bank et al.(2020)Bank, Koenigstein, and Giryes].

Generally autoencoders consist of an encoder part, a decoder part as neural networks and a latent space in between. This way they learn iteratively the encoding and decoding process.

In the figure 2.2, a simple autoencoder with the encoder, decoder and the latent space

- Encoder: The input is compressed into a latent-space representation by this portion of the network. It can be compared to a data compression technique

with data-specific compression.

- Latent Space Or Compressed representation: This section of the network shows the decoder's compressed input. In figure 4.3, an example of the latent space representation from the pcb images data is shown.
- Decoder: The input is rebuilt by this section of the network using the latent space representation. It can be compared to a data decompression technique, however the decompression is tailored to the data the network was trained on.

2.4.2 Variational Autoencoders

According to [Altosaar(2023)]; A variational autoencoder in the language of neural networks is made up of an encoder, a decoder, and a loss function. Since the latent space structure is the main source of the autoencoder's problems, it is not very effective at producing new pictures. Because of the latent spaces discontinuity, interpolation is difficult. There are enormous gaps between the clusters where encoded vectors are organized into clusters according to various data types. The decoder frequently produced nonsense output while creating a fresh sample if the selected point in the latent space was devoid of any data. Therefore came Variational autoencoder into existence in 2013 [OpenCV(2023)]. The techniques used in the variational bayesian and graphical models served as inspiration for the variational autoencoder. The foundation of VAE is Bayesian inference; as a result, it seeks to model the underlying probability distribution of data in order to sample fresh data from it [OpenCV(2023)]. In the figure 2.3 a Variational Autoencoder is shown.

The essential difference between VAE and a regular autoencoder is this fact: their latent spaces are continuous by design, making random sampling and interpolation simple. This trait is what makes VAE so valuable for generative modeling [OpenCV(2023)].

In the illustration 2.3, the encoder receives the input picture and produces two latent variables, Z_μ and Z_σ, which represent the distribution's parameters and are learned during training. In VAE, we have two latent variables Z_μ and Z_σ from which you sample a latent-vector Z rather than immediately producing a latent-space Z that is not required to follow any distribution. The sampled latent-vector

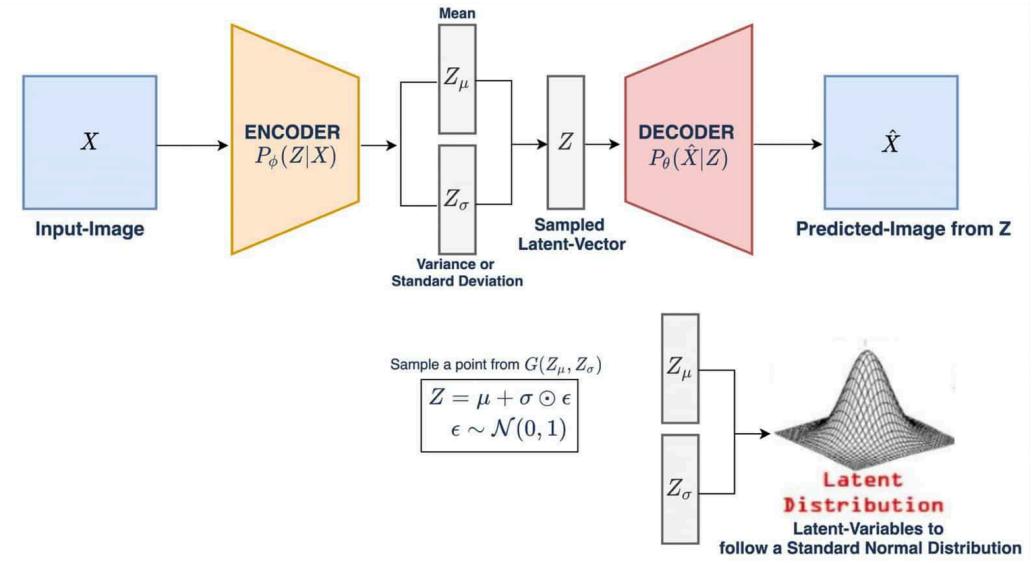

Figure 2.3: Variational Autoencoder
[OpenCV(2023)].

Z is also known as a sampling-layer since it draws data from a multivariate Gaussian $G(Z_\mu, Z_\sigma)$, where Z_μ and Z_σ stand for mean and variance, respectively. We get the anticipated picture $\hat{X}$ by passing the sampled vector Z to the decoder.The variables Z_μ and Z_σ must be trained such that they are close to zero and one, respectively, in order for our latent vector Z to follow a conventional normal distribution [OpenCV(2023)].

2.4.3 Arteficial Neural Networks(ANN)

Artificial Neural Networks (ANNs) are computational processing systems of which are heavily inspired by way biological nervous systems (such as the human brain) operate. ANNs are mainly comprised of a high number of interconnected computational nodes (referred to as neurons), of which work entwine in a distributed fashion to collectively learn from the input in order to optimise its final output [O'Shea and Nash(2015)].

2.4.4 Convolutional Neural Networks(CNN)

The Convolutional Neural Network (CNN) is one of the most amazing ANN architectural types. CNNs are typically employed to tackle challenging image-driven pattern recognition problems and, thanks to their accurate yet straightforward architecture, provide a streamlined way to get started with ANNs [O'Shea and Nash(2015)].

The field of computer vision has been significantly influenced by these convolutional neural network (CNN) which are neural network architectures [Heaton(2020)].

The learning process of CNN depends heavily on the arrangement of the input array's elements. As opposed to CNNs, many neural networks handle their input data as an extended vector of values, hence the placement of the input features within this vector is unimportant. For these kinds of neural networks, after the network has been trained, the order cannot be changed [Heaton(2020)].

The inputs are organised into a grid by CNN network. Concerning images, this setup worked effectively since pixels near to one another are essential for one another. It matters how the pixels are arranged in an image. An appropriate illustration of this kind of order is the human body. We are used to seeing eyes close together due to the facial structure [Heaton(2020)].

2.4.5 K Nearest Neighbors (KNN)

If someone has not much information about the distribution of the data, K nearest neighbor (KNN) classification should be one of the first options considered. It is one of the most fundamental and basic classification techniques. The necessity to perform discriminant analysis when accurate parametric estimates of probability densities are unidentified or difficult to ascertain led to the development of K-nearest-neighbor classification [Scholarpedia(2009)]. Fix and Hodges developed a non-parametric technique for pattern categorization in a 1951 study from the US Air Force School of Aviation Medicine that was never published. This technique is now known as the k-nearest neighbour rule [Fix and Hodges(1989)]. KNN is built on a distance function, which calculates how different or similar two instances are. The distance function is frequently the conventional Euclidean distance d(x,y) between two instances of x and y, defined as follows [Jiang et al.(2007)Jiang, Cai, Wang, and Jiang].

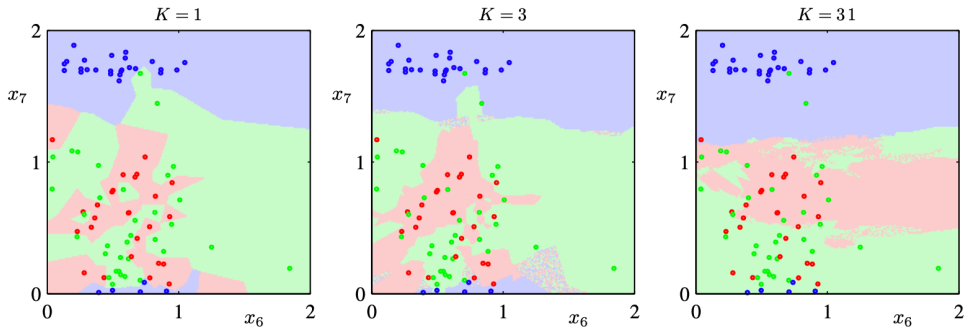

Figure 2.4: Is an example of the plot of 200 data points from an oil data set, in which the red, green, and blue colours represent the ”laminar,” ”annular,” and ”homogeneous” classes, respectively, shows values of x6 plotted against x7. The K-nearest-neighbor algorithm's categorization of the input space for various K values are also displayed.

[Bishop and Nasrabadi(2006)]

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (a_i(x) - a_i(y))^2} \tag{2.1}$$

The Euclidean distance between the sample being tested and the designated training samples is frequently used as the basis for the k-nearest-neighbor classifier. Let n be the sum of the number of input samples $(i = 1, 2, \ldots, n)$ and p be the entire number of features $(j = 1, 2, \ldots, p)$ and let x_i be a sample of the input with p features $(x_{i1}, x_{i2}, \ldots, x_{ip})$. The formula for the Euclidean distance of instance x_i and x_l $(l = 1, 2, \ldots, n)$ is

$$d(x_i, x_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + \ldots + (x_{ip} - x_{lp})^2} \tag{2.2}$$

[Scholarpedia(2009)].

2.4.6 Random Forests

Leo Breiman and Adele Cutler are the creators of the widely used machine learning technique known as random forest, which mixes the outcomes of various decision trees to produce a single outcome. Its widespread use is motivated by its adaptability and usability because it can solve classification and regression issues [IBM(2023)]. Random forests are made of decision trees. so it makes it

easier to understand if I introduce the decision trees first. Decision trees were among the first algorithms for statistics to be utilized in computer format following the introduction of electrical circuitry to digital calculations in the latter decades of the twentieth century, which are universal prediction and classification processes [De Ville(2013)]. In simple words, a decision tree creates an assumption, determines if it is accurate or untrue, and then produces a decision. The most widely used criterion for splitting are ”gini” for the Gini impurity as well as ”entropy” for the information gain, both of which have mathematical expressions as: The following formula calculates the entropy considering a dataset with classes N [Sarker(2021)].

$$E = -\sum_{i=1}^{N} p_i \log_2 p_i \tag{2.3}$$

and the Gini impurity is calculated as as follows: [Ansari(2022)].

$$Gini(t) = 1 - \sum_{i=1}^{N} P\left(\frac{i}{t}\right)^2 \tag{2.4}$$

The P denotes the class ratio at the ith node. Gini impurity gets an upper limit of 0.5, indicating the worst case scenario, but a lowest value of 0 indicating the best case scenario [Ansari(2022)].

Figure 2.5 shows an example of a decision tree. Using prior knowledge from the training data, Decision Tree is a common classification as well as prediction technique within supervised machine learning that may be used to label or categorise the provided data [Patil and Kulkarni(2019)].

Numerous classification trees are grown in Random Forests. Place the given input vector down each tree in the forest to classify an unfamiliar item using the input vector. Each tree makes a classification, which we refer to as the class the tree ”votes” for. Among every trees within the forest, the categorization with the highest votes is chosen by the forest [Breiman(2015)].

In order to decide on a general classification over the given collection of inputs, the random forest approach combines numerous random tree classifications. Each vote cast by a machine learner is typically given equal weight. Later works by Breiman extended this technique to support both weighted and unweighted voting. The classification with the most votes is the one the

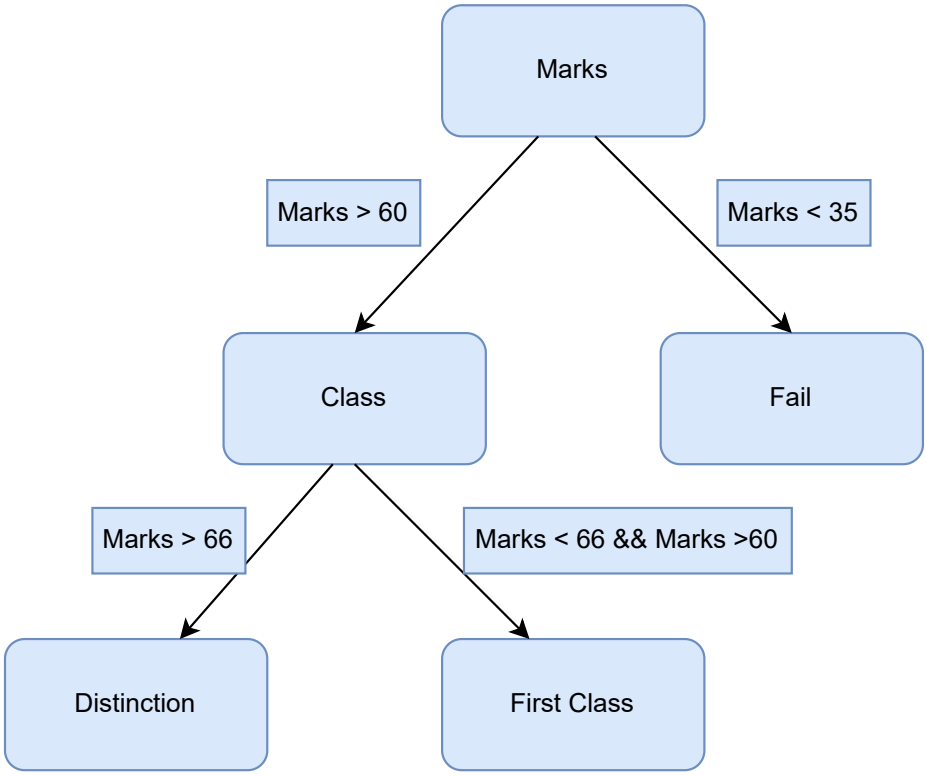

Figure 2.5: An example of a Decision Tree
[Somvanshi et al.(2016)Somvanshi, Chavan, Tambade, and Shinde].

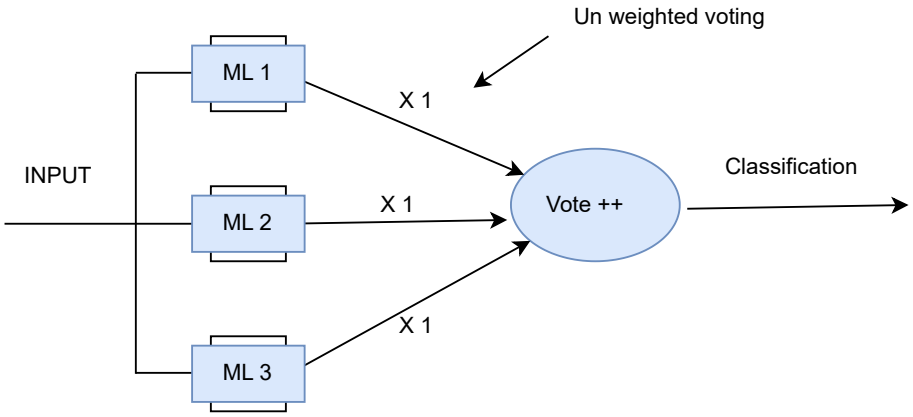

Figure 2.6: Random Forest Algorithm

.

forest choose [Livingston(2005)]. An illustration of the random forest algorithm(unweighted) may be seen in Figure 2.6

Supposing that we have been given a training sample of random variables that are independent and spread as an independent prototype pair (X, Y) with the formula: $\mathcal{S}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$. The objective is to create an estimate of the function m_n using the data set $\mathcal{S}_n$ and the formulas $m_n : \mathcal{D} \to \mathbb{R}$. In this sense, the regression function estimate m_n is said to be *(mean squared error)* consistent if

$$\mathbb{E}\{m_n(X) - m(X)\}^2 \to 0 \quad \text{as} \quad n \to \infty \tag{2.5}$$

(The expectation is assessed over the sample $\mathcal{S}_n$ and the variable X) [Biau and Scornet(2016)].

Chapter 3

Related Work

3.1 Introduction

Defect detection in industrial images is a critical task that helps to ensure product quality and maintain efficiency in the manufacturing process. Various techniques have been proposed to perform defect detection, including image processing techniques, machine learning techniques, and deep learning techniques.

In this literature review, we focus on deep learning techniques, specifically autoencoders, viariational autoencoders, denoising autoencoders and generative adversarial networks (GANNs), for defect detection in industrial images [Bai et al.(2014)Bai, Fang, Lin, Wang, and Ju].

3.2 Defects in PCB images

A printed circuit board, or PCB, is a crucial component in electrical equipment as it transports various parts. The quality of the PCB directly affects the performance of the equipment. The industry has widely adopted automated optical inspection (AOI) based on machine vision to improve efficiency and reduce fatigue associated with manual detection.

As PCBs become more complex, identifying and categorizing defects has become increasingly challenging [Huang and Wei(2019)].

Machine learning techniques have been now in use for a while to detect different kinds of defects some have used them to detect defect in fabric [Mei et al.(2018)Mei, Wang, and Wen], some have used them to detect defects in sur-

face images [Bhatt et al.(2021)Bhatt, Malhan, Rajendran, Shah, Thakar, Yoon, and Gupta], and even some have used them to detect weld defects based on digitised images [Nacereddine et al.(2005)Nacereddine, Zelmat, Belaifa, and Tridi] and some in crack detection of concretes structures [Islam and Kim(2019)] by using various deep learning methods such as transfer learning, convolutional neural networks, autoencoders and generative adversarial networks.

Some of these papers like [Mei et al.(2018)Mei, Wang, and Wen] employ deep learning methods specifically a multi scale convolutional denoising autoencoder network model due to the diversity and complexity of the defects in fabric images.

They developed a model to learn the characteristics of defective and non defective fabrics using a data set of fabric images with and without imperfections and the multi-scale architecture that the CAE network was built with enables it to learn textrue patterns at many scales and capture more substantial features for defect detection [Mei et al.(2018)Mei, Wang, and Wen].

In this paper they underwent evaluation using a benchmark dataset and showed promising accuracy and F1-score results even though they have mentioned challenges like there being many distinct textiles, each of which typically has unique set of properties which makes it hard to create universal algorithms suitable with different texture types [Mei et al.(2018)Mei, Wang, and Wen].

Another challenge they have mentioned is the variety of categories and traits associated with the defects in fabric in general. There exist already over 70 types of fabric defects that have been established by the textile industry.The last challenge mentioned is the difficulty for businesses to gather significant numbers of fabric samples, especially some unusual varieties, which cause data imbalances or outright failures for some classic supervised algorithms [Mei et al.(2018)Mei, Wang, and Wen].

Another article that used a unique module referred to as group pyramid pooling. which is designed to improve the detection of PCB defects is the the article by [Tang et al.(2019)Tang, He, Huang, and Yang].

The article creates or collects a new dataset of PCB images with 6 different categories of defects and annotates those six sorts of typical defects.Then they apply Max pooling convolutional backbone followed by a novel group pyramid pooling which they used to effectively incorporate data from multiple resolutions and forecast PCB defects at different scales [Tang et al.(2019)Tang, He, Huang, and Yang].

Finally in-depth tests were performed to asses the suggested model.The [Tang et al.(2019)Tang, He, Huang, and Yang] claim that their approach can accomplish stat-of-the -art performance while consuming relatively small computing time and that future research will be made easier by the accessible DeepPCB dataset.

The [Huang and Wei(2019)] is a similar article which presented a generated PCB dataset that includes 1386 images of defects with 6 different types for use in detection, classification, and registration tasks.

The ting that motivated them for the creation of generated PCB dataset was because many researches rarely publish their dataset even though they have suggested several of their methods [Huang and Wei(2019)].

In this article by [Huang and Wei(2019)] In order to categorize the flaws, they presented a reference-based technique of inspection and trained an end-to-end convolutional neural network.

Their method performs better on their dataset than traditional approaches that call for pixel-by-pixel processing by first locating the defects and then classifying them using neural networks. according to them these datasets are available online [Huang and Wei(2019)].

To increase the data´s size they implemented data augmentation methods and used dense shortcuts from densenet instead of just stacking convolutional layers in order to improve learning. having used these they claim that they were able to attain great precision [Huang and Wei(2019)].

Another PCB defect detection method that uses deep learning methods is the paper by [Wu et al.(2021)Wu, Ge, Zhang, and Zhang].

This document setts off in order to guarantee high output and reduce labor costs by using two different types of target detection networks to identify and categorize pcb defects [Wu et al.(2021)Wu, Ge, Zhang, and Zhang].

The paper utilises two kinds of models which are the SSD and FPN models in two different kinds of PCB images.

According to [Wu et al.(2021)Wu, Ge, Zhang, and Zhang], On these datasets with various distributions, both of these models have produced positive outcomes.

According to their findings, the FPN effectively detects the defects on both PCB datasets.The value of the f1 score on the PCB dataset using SSD is only 54.4%, thus the detection effectiveness of SSD is somewhat poorer than that of FPN [Wu et al.(2021)Wu, Ge, Zhang, and Zhang].

PCB images were also used in a different yet interesting defect detection method using the denoising convolutional autoencoder by [Khalilian et al.(2020)Khalilian, Hallaj, Balouchestani, Karshenas, and Mohammadi].

This article attempts to restore the original image from a corrupted one. The model was created to replace the defective components after being trained on defective PCBs [Khalilian et al.(2020)Khalilian, Hallaj, Balouchestani, Karshenas, and Mohammadi].

Here they point out that in the past, image processing techniques were used to automate the visual examination of PCBs, but machine learning approaches, including neural networks and convolutional neural networks, have now taken their place [Khalilian et al.(2020)Khalilian, Hallaj, Balouchestani, Karshenas, and Mohammadi].

They also mention that unsupervised neural networks using autoencoders are effective at obtaining outputs by extracting features from data sources [Khalilian et al.(2020)Khalilian, Hallaj, Balouchestani, Karshenas, and Mohammadi].

Here they removed the noise first from the PCB images before the noise-free images were sent into a deep learning method known as Convolutional Autoencoder network for training. Then by reconstructing the input images and comparing them with the original images to find changes, the trained network was then utilized to detect defects in newer, untested PCB images [Khalilian et al.(2020)Khalilian, Hallaj, Balouchestani, Karshenas, and Mohammadi].

This way they were able not only find and located the problems of every description but also were able to fix them. They identified the defective components by deducting the output of the repaired system from the input [Khalilian et al.(2020)Khalilian, Hallaj, Balouchestani, Karshenas, and Mohammadi].

According to this paper the experimental results show that this approach, when compared to current state-of-the-art works, obtained high accuracy in the detection of defective PCBs which is 97% [Khalilian et al.(2020)Khalilian, Hallaj, Balouchestani, Karshenas, and Mohammadi].

The paper by [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard] also presented another interesting work in this field.

According to this paper,they were set to find the solution to the shortage of publicly available data sets for printed circuit boards (PCB-AVI), within the field of Automated visual inspection. This due to the reason that there aren't many publicly available data-sets of this kind and thus the developments and capabil-

ities within this field aren't well understood. As a result they suggested a publicly accessible dataset called "FICS-PCB" in order to support the development of reliable PCB-AVI algorithms [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].

They also mention that their suggested dataset comprises tough instances related to illumination, image scale, and image sensor characteristics and includes 9,912 images of 31 PCB samples and 77,347 annotated components [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].

The dataset contains challenging instances related to three different variables: illumination, imagine scale, and image sensor. This collection includes 77,347 annotated components and 9,912 images of 31 PCB samples [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].

It is also mentioned that, they used the following methodologies,

- On the suggested dataset,experiments are run to assess performance changes. They use cutting-edge feature engineering and deep learning techniques to classify PCB components on the variable subsets [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].

- Since image matching techniques take a lot of time and are sensitive to image changes, they are not used in the research [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].

- RGB color As part of the feature engineering process,the Fourier shape and fusion of color and edge characteristics were used [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].

- The R,G,and Bcolor channel values are retrieved from a 10 10 pixel square in the center of the component body and given into a Naive Bayes classifier as RGB color features [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].

- As it was applied,the Fourier descriptor is employed along with Canny edge detection and watershed segmentation [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].

- The sixth coefficients are used to incorporate enough shape features after applying the discrete complex Fourier transform to generate the Fourier

coefficients [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].

- The multi-layer perceptron is then fed with these coefficients (MLP) [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].
- It uses feature fusion, which joins feature vectors for color and shape [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].
- The training of an MLP for the identification of PCB components uses the concatenation of three features [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].
- The potential of applying deep learning for PCB component categorization was investigated using two deep neural networks, AlexNet and Inception-v3 [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].

This research also gives the following as some of the challenges faced;

- These widely accessible datasets are sizable in terms of size, but they do not replicate the broad variation in real-world situations that can present difficulties for PCB inspection performance [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].
- On private datasets, the performance of existing PCB-AVI approaches has been encouraging [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard]. and
- The necessity for a new dataset is driven by the text's emphasis on the difficulties in creating effective PCB Automatic Visual Inspection techniques due to the great heterogeneity in PCB elements, surfaces, and different types of imaging [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].

As results it is mentioned that the Baseline performance is above 97%, Low accuracy in the Illumination variation(less than 60%), dropped classification accuracy in the scale variation and a drop by 35% was seen in the Image sensor variation which showed potential for future development [Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard].

The article by [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang] also presents a deep learning based detector for the rapid and precise detection of surface defects in printed circuit boards(PCBs).

The YOLOv4 was enhanced in terms of its neck/prediction network's activation function and backbone network. The work suggests YOLOv4-MN3 as a low-cost DL-based detector to address the problems with large parameters and computational expense in existing models [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].

The article implemented the following methodologies;

- In general, computer vision and deep learning approaches are employed, and in this case, the method was developed based on the state-of-the-art YOLOv4 one-stage detector [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].
- It entails three steps: dataset construction, model training, and performance assessment [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].
- A specially created image acquisition device first gathers all of the PCB defect images, and then the augmentation and annotation labeling are carried out. After that, YOLOv4 is adjusted and trained using a unique dataset. The effectiveness of the proposed YOLOv4-MN3 and other SOTA detectors is assessed and compared in the last section [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].
- For this study, the lightweight network MobileNetV3 was substituted for the CSPDarknet53 in YOLOv4 to create the cost-effective detector YOLOv4-MN3. MobileNetV3 builds feature mappings for each layer using depthwise separable convolution [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].
- The squeeze and excitation attention module of MobileNetV3 is added to the MobileNetV2 bottleneck in order to increase the detection accuracy [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].
- To increase the precision of detection, MobileNetV3 updates the Swish activation function [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].

- Due to its ability to achieve high detection speed and accuracy at the same time, experimental findings attest to MobileNetV3's usefulness and superiority [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].
- To choose the most appropriate activation function based on how well it performed during training and detection on the specific dataset, multiple activation functions' features were examined, and comparison studies between them were run [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].
- The activation function for the neck/prediction network is the sole one chosen in this study as MobileNetV3 has optimized its activation function [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].

Here, the paper claims that their results showed that when compared to other models, YOLOv4-MN3 had the greatest detection accuracy, fastest speed, and lowest Madds [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].

For useful DL-based models, however, the complexity and diversity of genuine industry backgrounds, defect categories, and morphologies must be constantly updated [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].

Furthermore they mention that, YOLOv4-MN3's extensive use of manually labeled samples necessitates the investigation of unsupervised or semi-supervised techniques [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].

According to the paper The enhanced detector outperformed existing state-of-the-art (SOTA) detectors when tested using a tailored dataset gathered from PCB manufacturing, achieving a mean average precision (mAP) of 98.64% at 56.98 frames per second (FPS) [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].

It was also reported that YOLOv4's multiply-accumulate operations and parameter space reduction.It is also clear that MobileNetV3 outperformed all other backbone networks in terms of mAP [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].

The value of mAP obtained by MobileNetV3 is higher than that of VGG16, Resnet50, Darknet53, CSPDarknet53, and MobileNetV2 by 1.26%, 2.16%, 1.65%, 0.77%, and 2.31%, respectively [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].

Within the six backbone networks, MobileNetV3 had the greatest F1 score, improving above VGG16, Resnet50, Darknet53, CSPDarknet53, and MobileNetV2

by 2.66%, 5.16%, 2.83%, 0.66%, and 4.50%, respectively [Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang].

Table 3.1 shows a summary of the above mentioned important research articles reviewed in this chapter. AS it can be seen from the table the majority of the articles used PCB image datasets for their methods in their studies except Mei et al.(2018) that used Fabric Images.

Author/Article	Method	Dataset Used	Accuracy
Mei et al.(2018)]	Conv Denoising AE	Fabric Images	80%
Tang et al.(2019)	Group pyramid pooling	Deep PCB	
Huang and Wei(2019)	end-to-end conv neural net	PCB	99%
Wu et al.(2021)	Convolutional neural net	PCB	
Khalilian et al.(2020)	Denoising conv AE	PCB	97.5%
Lu et al.(2020)	Deep neural net	FICS-PCB	
Liao et al.(2021)	YOLOv4-MN3	PCB	98.64%

Table 3.1: A summary of the related work

Chapter 4

Dataset

4.1 Dataset in machine learning

A dataset in machine learning is a set of data which is used to train a model. A dataset serves as an example for the algorithm used for machine learning to learn how to generate predictions.

The following are examples of typical data types: video,audio,image,text and numeric data [Javaid(2023)]. These datasets are mostly used to train or test the model.

4.2 PCB Image datasets

PCB data sets are collections that have data created especially for testing and refining defect detection algorithms [Tang et al.(2019)Tang, He, Huang, and Yang].

4.3 DEEP PCB

DEEP PCB is A dataset of 1,500 image pairings includes locations of the six most prevalent PCB defects: open, short, spur, mousebite, pinhole, and spurious copper. Each image pair includes a free of defects template image and an equivalent testing image [Tang et al.(2019)Tang, He, Huang, and Yang].

4.4 Dataset Description

The linear scan CCD used to capture each image in this dataset has a pixel density of roughly 48 pixels per 1 millimeter. As described above, the non-defective template images are personally examined and cleaned from the sample pictures.

The templates and the tested image's original dimensions are roughly 16k by 16k pixels. They are then divided into numerous 640 × 640 pixels sub-images using a cropping process, then aligned using template matching methods.

The next step is to carefully choose a threshold to use binarization to prevent lighting disturbance. The pre-processing algorithms can vary depending on the specific PCB fault detection algorithms. Nevertheless, high-accuracy PCB defect recognition and classification methods often use picture registration and thresholding approaches [Tang et al.(2019)Tang, He, Huang, and Yang].

Examples of these image datasets are shown in figures 4.1 and 4.2 as defective images and non defective images respectively below them to give an idea of how they look like and show their difference.

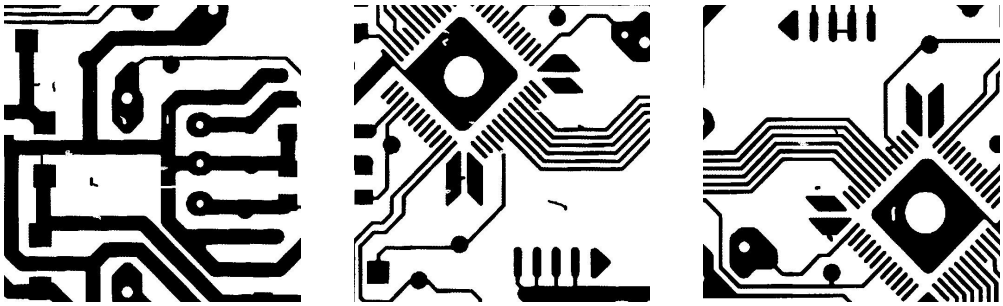

Figure 4.1: Defective Images

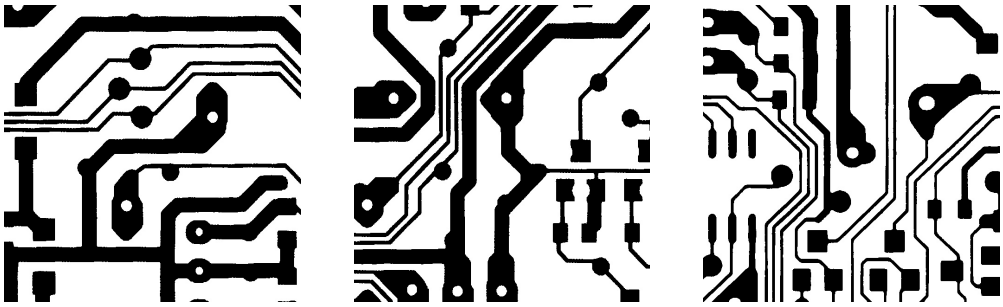

Figure 4.2: Non-Defective Images

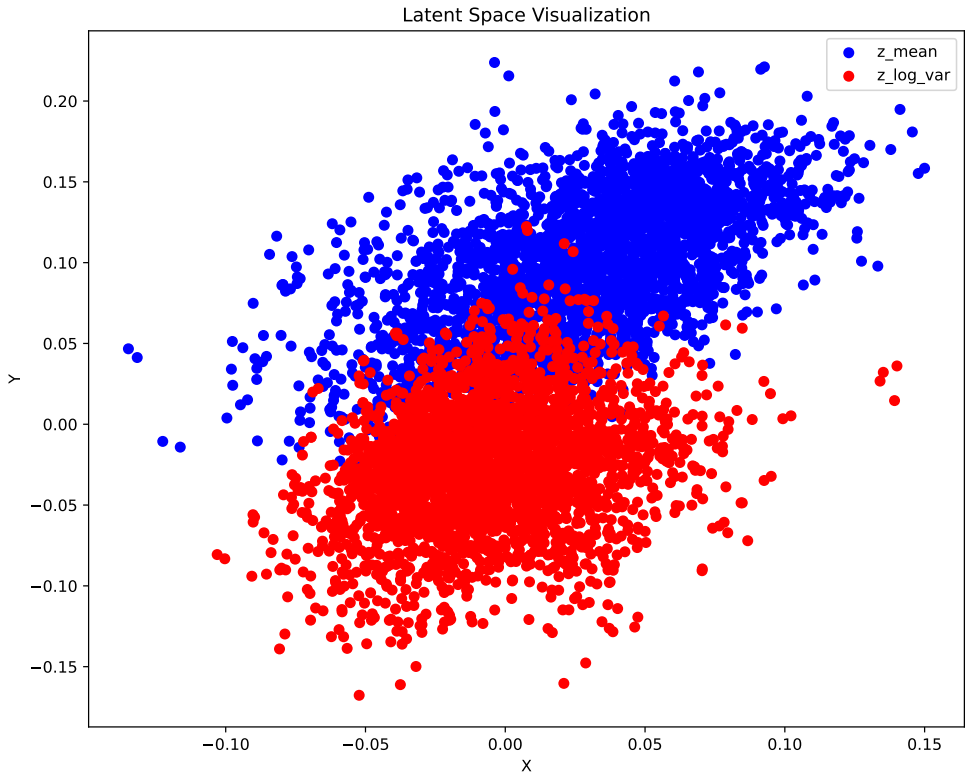

Figure 4.3: Latent space representation from the PCB images dataset

4.5 Preprocessing of the PCB image datasets

Here these gray scale image datasets have been resized to a reasonable degree to make it easier for the machine to work on.

Normalisation was also applied to scale the values of the image pixel values, i.e from [0,255] to the range of [0,1] or [-1,1] to help in the convergence of the training process.

Data augumentation technique was also used in some cases to expand the number of the dataset by making different altered copies of the image datasets.

Figure 4.3 shows a scatter plot that visualizes the distributions of latent space representations learned by an autoencoder.The autoencoder model learns to

compress the information of each image into a lower-dimensional latent representation.

The z_mean represents the mean value of the latent variables and is shown with the blue colour dots in figure 4.3. The z_log_var on the other hand represents the logarithm of the variances of the latent variables. This is represented by the points with red color in Figure 4.3.

Chapter 5

Methodology

- A literature review was carried out on the different kinds of defect detection methods that involved autoencoders.

  This was followed by a study of the autoencoders, how they work, how to construct and implement them along side the use of python in jupyterLab and different other libraries like Keras, tensorflow, Numpy, matplotlib,SciKit-Learn and others to work with the loading, reading, resizing, preprocessing, building the model architechture and compiling of the models.

- Integrates image processing and machine learning methods to detect defects in industrial images.

- Analyses various cutting-edge strategies for detecting defects, including:

  - Auto Encoder (AE)
  - Variational Auto Encoder (VAE)
  - K Nearest Neighbours
  - Random Forests

- Applies and evaluates these techniques on different classifiers.

- Primarily uses image datasets from Printed Circuit Boards (PCBs) that include defective and non defective images.

- Uses these PCB image datasets to train and test various models.

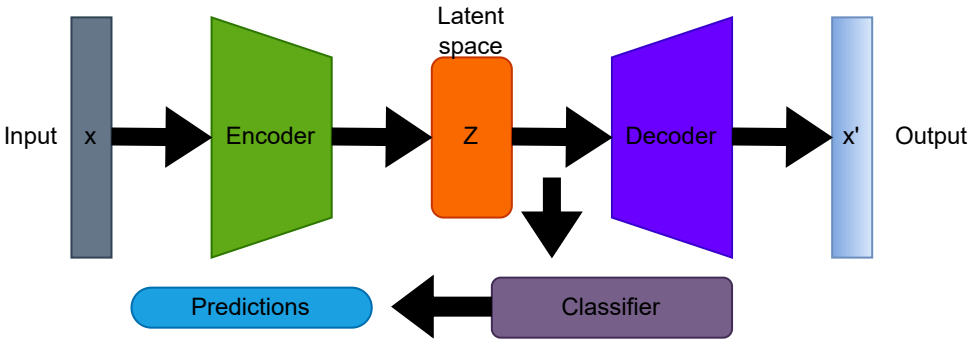

Figure 5.1: A chart that shows the methodology used in this thesis

.

- Extensively trains models using a wide range of hyper-parameters and architectures before testing on other datasets.

The Main Methodology implemented in this thesis is the process of Autoencoder. But instead of the conventional method where the input data gets compressed in the latent space and then again getting reconstructed to check the error loss between the original input and the reconstructed image to check whether the image is defective or non defective.

In the methodology which was applied here, the data was taken from the latent space and instead of it getting reconstructed, it is fed to the classifier with the data and then the classifier checks the defectiveness of the image from the data from the latent space.

Here Convolutional variational autoencoder, the K-Nearest neighbors classifier and the Random forest classifier are used. This procedure is shown in the flow chart in the Fig 5.1

The second one which is used here is the conventional method of using autoencoders. It can be seen in the Fig 5.2. This method contains the Encoder, Decoder and Latent space where the encoder extracts the compressed version from the input image because it learns characteristics from the input image and has fewer nodes in its output layer than in its input layer. The decoder uses this compressed representation as input and reconstructs the input image as output. Each output node represents a specific pattern of geometric shapes.[**?**]

As with all other neural network techniques, the autoencoder in the above-mentioned methodologies trains the system using images from the image data

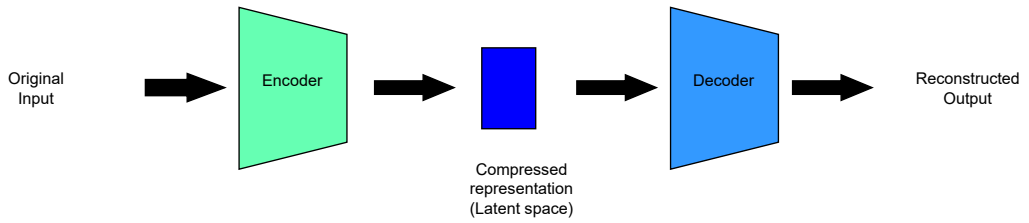

Figure 5.2: The conventional Autoencoder method

.

utilizing backpropagation techniques.

Each image's reconstruction error (which is also called loss function) is calculated; if it is significant, the model sees the image as defective. Else the image is non-defective. Additionally, a threshold is established above which an image is considered defective.

Chapter 6

Results and Discussion

The findings and a discussion of the thesis are presented in this chapter. Here, the findings are summarised, shown in several formats, including graphic and tabular ones, reviewed, and contrasted with previous works. Starting with the structure of the autoencoders, this section will explore through the training methods analyse the results of the models and finish with a discussion of the many aspects of the thesis.

Note: Here accuracy is taken as the main metric for evaluating performance as the image dataset we are using is fairly balanced between the defective and non defective images.

6.1 Structure of the Model

6.2 Autoencoder

As was mentioned earlier in chapter 4, the PCB images were resized and called as D_resized for the defective PCB images and N_resized for the non-defective PCB images. The images are labeled 1 for the defective and 0 for the non-defective.

They are also normalised to a scale of [0,1] to eliminate data quality problems,boost data analysis and lessen data duplication for a better training.

6.2.1 The encoder

The input layer of the encoder is initially created to handle gray scale image of dimentions (224 x 224). Two convolutional layers, each with 32 and 64 filters, are

used by the encoder. Each of those layers have a stride of 2, and batch normalisation comes after leaky ReLU activations. The structure then uses dense layers to generate a latent representation of the dimension latent_dim, which was set to 128. This is done after flattening the spatial data.

6.2.2 The decoder

From the latent space, the decoder recreates the original image.It begins by transforming the data back into a 2D format using a dense layer, and then utilises two transposed convolutional layers to restore the image back to its original 224 x 224 size. The table 6.1 shows A summary of the Autoencoder structure used.

Layer type	**Encoder**	**Decoder**
Input shape	224x224x1	128(latent_dim)
Conv2D(strides=2)	32 filters	
Activation	LeakyReLU(alpha=0.2)	
BatchNormalization	Yes	
Conv2D(strides=2)	64 filters	
Activation	LeakyReLU(alpha=0.2)	
BatchNormalization	Yes	
Flatten/Dense	Flatten + 128 nodes	Dense+56x56x64
Reshape		56x56x64
Conv2DTranspose		64 filters(strides=2)
Activation		LeakyReLU(alpha=0.2)
BatchNormalization		Yes
Conv2DTranspose		32 filters(strides=2)
Activation		LeakyReLU(alpha=0.2)
Output	128(latent_dim)	224x224x1

Table 6.1: Structure of the Autoencoder

6.2.3 Training

Using a learning rate of 0.001, the RMSprop optimizer was used to train the autoencoder while taking into account the mean square error(MSE) of the initial and reconstructed images. Using an early stopping mechanism,training was stopped if no progress was seen for 10 consecutive epochs in the validation loss. Training was done using a batch size of 32 across 50 epochs.

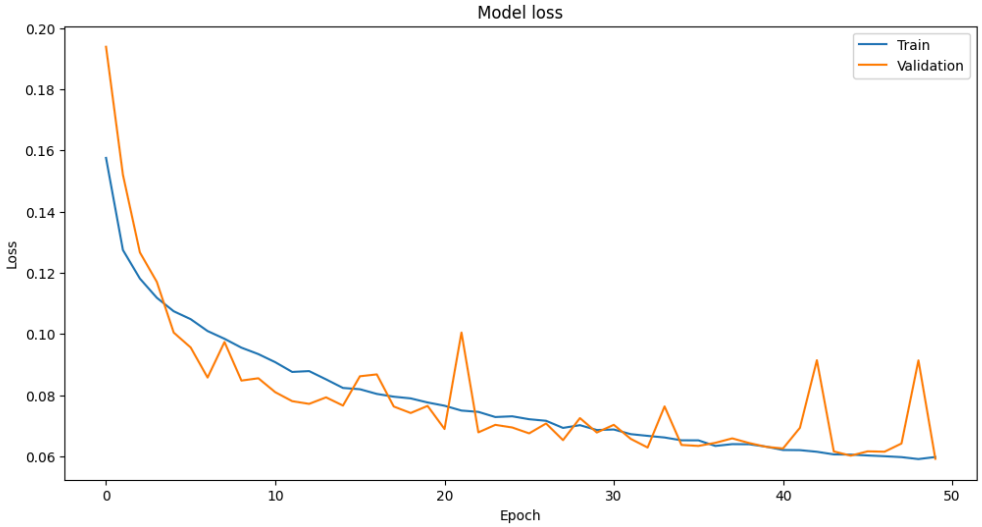

Figure 6.1: Loss plot for the Autoencoder model

.

Fig 6.1 shows the training and validation loss for the Autoencoder model. The training configurations of the Autoencoder are given in the table 6.2

Parameter	**Value**
Optimizer	RMSprop
Learning Rate	0.0001
Loss Function	Mean Square Error(MSE)
Epochs	50
Batch Size	64
Early Stopping	Yes(patience=10)

Table 6.2: Training configurations of the AE

6.3 Structure of the Model

6.4 Variational Autoencoder(VAE) with a KNN classifier

After labelling the defective images(D_images as 1 and N_images as 0) and normalizing the pixel values to be between 0 and 1, the latent space data is split in to

training and testing sets as this will be applied or fed to the classifier.

6.4.1 Encoder

A 224 by 224 grayscale image is fed into the encoder. Dense layers are applied next, after which 2D convolutional layers are applied. Then, z_mean and z_log_var are the output.

6.4.2 Latent Space

The latent representation (z_mean) of the images from the VAE is used to train a KNN classifier in this model. The images will then be categorised as defective or non-defective so that the performance of the KNN model may be evaluated.

Table 6.3 shows the latent space Loading and data splitting process

Parameter	**Input**	**Output**	**Aim**
Loading z_mean and z_log_var	output_folder path	z_mean and z_log_var	load pre calculated latent space structures
Data Spliting	z_mean labels	trainz_mean,testz_mean, train_labels,test_labels	divide data in to training and testing sets

Table 6.3: Latent space loading and data splitting

6.4.3 Decoder

The values from the latent space are sent into the decoder. Utilising dense layers and transpose convolution layers, it transforms them again into an image.

Here, the model is produced by passing the encoded data in to the decoder

6.4.4 Loss and Compilation

Here, reconstruction loss and KL divergence loss are combined in the VAE loss.

The Adam optimizer along with the unique VAE loss are used to build the model.

6.4.5 VAE + KNN Image Classification

With the help of the VAE's encoder, an image is uploaded and its latent representation is extracted.

The KNN classifier is then applied to decide whether or not this image is defective.

6.5 Structure of the Model

6.6 Random Forest

Random forest has a simple architecture in comparison to Convolutional Neural net but the combination of multiple decision trees can in some cases give it an extra strength to perform better.

The following table 6.4 shows the Random Forest Structure

Component	**Description**
Base Model	Decision Ttree
Number of Trees	n_estimatores = 100
Bootstrap Samples	Yes
Max Features	Default
Criterion	Gini impurity
Tree depth	not limited
Min samples split	2
Min samples leaf	1

Table 6.4: Random Forest Structure

6.6.1 Training

The Random forest is traied on the latent representation(train_z_mean) in a similar fashion to the KNN and the convolutionla neural network classifier. Training of the VAE and the Neural network classifier which are present along with the Random Forest classifier is already mentioned in the other sections, so no need to repeat it here.

6.7 Structure of the Model

6.8 Convolutional Neural Network

Following loading and data preparation, data augmentation is used to artificially expand the total amount of the data set used for training by altering the dataset's existing images. This is accomplished through rotation, width-and-height shifting, shearing, zooming, and horizontal flips.

Layer Type	**Parameters**	**Activation Function**
Conv2D	32 filters, kernel size 3x3	ReLU
Conv2D	64 filters, kernel size 3x3	ReLU
Maxpooling2D	pool size 2x2	
Dropout	0.5	
Conv2D	128 filters, kernel size 3x3	ReLU
Conv2D	256 filters, kernel size 3x3	ReLU
MaxPooling2D	pool size 2x2	
Dropout	0.5	
Flatten		
Dense	128 units	ReLU
Dense	64 units	ReLU
Dense	1 unit	Sigmoid

Table 6.5: CNN classification model with data Augumentation

Table 6.5 shows CNN classification model with data augmentation

Layer Type	**Parameters**	**Activation Function**
Conv2D	32 filters, kernel size 3x3	ReLU
Conv2D	64 filters, kernel size 3x3	ReLU
Flatten		
Dense	64 units	
Dense	1 unit	ReLU

Table 6.6: CNN classification model without data Augumentation

6.8.1 VAE Encoder

The table 6.7 here, provides the architecture of the models in terms of their layers, the parameters of each layer, and the activation functions used.

Multiple convolutional layers in the encoder decrease the spatial dimensions simultaneously increasing the depth. The data is flattened and fed into dense layers in this manner until the z_mean and z_log_var are produced.

Layer Type	**Parameters**	**Activation Function**
Conv2D	32 filters, kernel size 3x3, stride 2x2	ReLU
Conv2D	64 filters, kernel size 3x3, stride 2x2	ReLU
Flatten		
Dense	16 units	ReLU
Dense(z_log_var)	latent_dim units	
Dense(z_log_var)	latent_dim units	

Table 6.7: VAE in CNN encoder structure

6.8.2 VAE Decoder

In a similar manner the table 6.8 describe the architecture of the models in terms of their layers, the parameters of each layer, and the activation functions used.

Inverting what the encoder does is what the decoder does. Dense layers are used to upsample the data starting from the latent vector. Conv2DTranspose layers are then used to decrease depth and increase spatial dimensions. After that, the data is rebuilt in the manner that it originally was obtained.

6.8.3 Analysis

In all the cases,the mean square error (MSE) between the initial and reconstructed image, which is the reconstruction error in this case was calculated to check whether the images is defective or non-defective.in the case of the autoencoder only, the ninety-five percentile of the reconstruction errors for the validation set was chosen as the error's threshold value. In the other cases a fixed threshold value was given through several trial and error method.

Layer Type	Parameters	Activation Function
Dense	56 x 56 x 64 units	ReLU
Conv2DTranspose	64 filters, kernel size 3x3, stride 2x2	ReLU
Conv2DTranspose	32 filters, kernel size 3x3, stride 2x2	ReLU
Conv2DTranspose	1 filter, kernel size 3x3	Sigmoid

Table 6.8: VAE in CNN Decoder structure

Again in all the cases, the model classifies the image showing a mean square error(MSE) greater than this threshold as a defective image.

To sum up the results, The VAE using convolutional neural net showed the highest accuracy in the defect detection of the PCB images which is 98.5%.

Model	Accuracy %
Autoencoder	53
KNN(using latent space)	60
RF(using latent space)	49.5
CNN(using latent space)	98.5

Table 6.9: The best achieved accuracy results for the different models

Table 6.9 shows the best achieved accuracy results by the different models. Different hyper parameter tuning methods where implemented here. To mention some, Adjusting the learning rate,data augumentation, changing the optimizer, modifying regularization(like the L1,L2) and others. Most of them had visible effect on the VAE with CNN, but the rest responded poorly. Compared to table 3.1 in chapter 3 The CNN using the latent space in this thesis performed reasonably well. As with the other models that didn't perform well in this thesis,there there is always room for improvement. One of the limitations was the capacity of the computing device used, which was struggling with the volumes of data used and taking hours to run.

Chapter 7

Conclusion

This thesis's main objective was to use PCB images for defect detection to compare the use of autoencoders in a conventional way versus using the latent space before the data in the latent space gets decoded. Different classification methods like KNN, RF, and CNN were fed with the data to see if their use of the latent space could help correctly identify defects in PCB images. The results showed that, at this moment, the application of the autoencoders by using the latent space to feed to other classifiers is practical with CNN. Even though the other methods did not achieve high accuracy in this study in detecting the PCB defects, It is evident that autoencoders have high potential, and approaching them with the right tools and other methods and ideas can prove rewarding in the future. Moreover, in the future, implementing the principles tried in this thesis with autoencoders in different datasets rather than PCB images can be an acceptable idea, and exploring other machine learning techniques that can go well along with autoencoders like Generative Adversarial Networks(GANS) can have satisfactory outcome.

Appendix A

Instructions to Compile and Run System

The relevant Python code for the tasks performed in this thesis is present on GitHub repository named ”Defect_detection”and can be accessed using this link https://github.com/SYohannesB/Defect_detection to the repository.

Bibliography

[Birchfield(2023)] Stan Birchfield. *Image procecssing and analysis*. Cengage Learning, 2023.

[Baldi(2012)] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012.

[Bank et al.(2020)Bank, Koenigstein, and Giryes] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *arXiv preprint arXiv:2003.05991*, 2020.

[OpenCV(2023)] OpenCV. Variational autoencoder in tensorflow, 2023. URL `https://learnopencv.com/variational-autoencoder-in-tensorflow/`. Accessed: 2023-06-06.

[Altosaar(2023)] Jaan Altosaar. What is a variational autoencoder? deep learning, in simple terms, 2023. URL `https://jaan.io/what-is-variational-autoencoder-vae-tutorial/`. Accessed: 2023-06-06.

[O'Shea and Nash(2015)] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[Heaton(2020)] Jeff Heaton. Applications of deep neural networks with keras. *arXiv preprint arXiv:2009.05673*, 2020.

[Scholarpedia(2009)] Scholarpedia. K-nearest neighbor, 2009. URL `http://scholarpedia.org/article/K-nearest`$_n eighbor.[Online; accessed 28-July-2023]$.

[Fix and Hodges(1989)] Evelyn Fix and Joseph Lawson Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, 57(3):238–247, 1989.

[Jiang et al.(2007)Jiang, Cai, Wang, and Jiang] Liangxiao Jiang, Zhihua Cai, Dianhong Wang, and Siwei Jiang. Survey of improving k-nearest-neighbor for classification. In *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, volume 1, pages 679–683, 2007. doi: 10.1109/FSKD.2007.552.

[Bishop and Nasrabadi(2006)] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[IBM(2023)] IBM. Random forest, 2023. URL `https://www.ibm.com/topics/random-forest`. [Online; accessed 28-July-2023].

[De Ville(2013)] Barry De Ville. Decision trees. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(6):448–455, 2013.

[Sarker(2021)] Iqbal H Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, 2(3):160, 2021.

[Ansari(2022)] Waqqas Ansari. Understanding the maths behind the gini impurity method for decision tree split, 2022. URL `https://analyticsindiamag.com/understanding-the-maths-behind-the-gini-impurity-meth` [Online; accessed 7-August-2023].

[Somvanshi et al.(2016)Somvanshi, Chavan, Tambade, and Shinde] Madan Somvanshi, Pranjali Chavan, Shital Tambade, and S. V. Shinde. A review of machine learning techniques using decision tree and support vector machine. In *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, pages 1–7, 2016. doi: 10.1109/ICCUBEA.2016.7860040.

[Patil and Kulkarni(2019)] Siddalingeshwar Patil and Umakant Kulkarni. Accuracy prediction for distributed decision tree using machine learning ap-

proach. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1365–1371. IEEE, 2019.

[Breiman(2015)] Leo Breiman. Random forests leo breiman and adele cutler. *Random Forests-Classification Description*, 106, 2015.

[Livingston(2005)] Frederick Livingston. Implementation of breiman's random forest machine learning algorithm. *ECE591Q Machine Learning Journal Paper*, pages 1–13, 2005.

[Biau and Scornet(2016)] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25:197–227, 2016.

[Bai et al.(2014)Bai, Fang, Lin, Wang, and Ju] Xiaolong Bai, Yuming Fang, Weisi Lin, Lipo Wang, and Bing-Feng Ju. Saliency-based defect detection in industrial images by using phase spectrum. *IEEE Transactions on Industrial Informatics*, 10(4):2135–2145, 2014.

[Huang and Wei(2019)] Weibo Huang and Peng Wei. A pcb dataset for defects detection and classification. *arXiv preprint arXiv:1901.08204*, 2019.

[Mei et al.(2018)Mei, Wang, and Wen] Shuang Mei, Yudan Wang, and Guojun Wen. Automatic fabric defect detection with a multi-scale convolutional denoising autoencoder network model. *Sensors*, 18(4):1064, 2018.

[Bhatt et al.(2021)Bhatt, Malhan, Rajendran, Shah, Thakar, Yoon, and Gupta] Prahar M Bhatt, Rishi K Malhan, Pradeep Rajendran, Brual C Shah, Shantanu Thakar, Yeo Jung Yoon, and Satyandra K Gupta. Image-based surface defect detection using deep learning: A review. *Journal of Computing and Information Science in Engineering*, 21(4), 2021.

[Nacereddine et al.(2005)Nacereddine, Zelmat, Belaifa, and Tridi] N Nacereddine, M Zelmat, Ss S Belaifa, and M Tridi. Weld defect detection in industrial radiography based digital image processing. *Transactions on Engineering Computing and Technology*, 2:145–148, 2005.

[Islam and Kim(2019)] MM Manjurul Islam and Jong-Myon Kim. Vision-based autonomous crack detection of concrete structures using a fully convolutional encoder–decoder network. *Sensors*, 19(19):4251, 2019.

[Tang et al.(2019)Tang, He, Huang, and Yang] Sanli Tang, Fan He, Xiaolin Huang, and Jie Yang. Online pcb defect detector on a new pcb defect dataset. *arXiv preprint arXiv:1902.06197*, 2019.

[Wu et al.(2021)Wu, Ge, Zhang, and Zhang] Xing Wu, Yuxi Ge, Qingfeng Zhang, and Dali Zhang. Pcb defect detection using deep learning methods. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 873–876. IEEE, 2021.

[Khalilian et al.(2020)Khalilian, Hallaj, Balouchestani, Karshenas, and Mohammadi] Saeed Khalilian, Yeganeh Hallaj, Arian Balouchestani, Hossein Karshenas, and Amir Mohammadi. Pcb defect detection using denoising convolutional autoencoders. In *2020 International Conference on Machine Vision and Image Processing (MVIP)*, pages 1–5. IEEE, 2020.

[Lu et al.(2020)Lu, Mehta, Paradis, Asadizanjani, Tehranipoor, and Woodard] Hangwei Lu, Dhwani Mehta, Olivia Paradis, Navid Asadizanjani, Mark Tehranipoor, and Damon L Woodard. Fics-pcb: A multi-modal image dataset for automated printed circuit board visual inspection. *Cryptology ePrint Archive*, 2020.

[Liao et al.(2021)Liao, Lv, Li, Luo, Zhu, and Jiang] Xinting Liao, Shengping Lv, Denghui Li, Yong Luo, Zichun Zhu, and Cheng Jiang. Yolov4-mn3 for pcb surface defect detection. *Applied Sciences*, 11(24):11701, 2021.

[Javaid(2023)] Shehmir Javaid. Quick guide to datasets for machine learning in 2023, 2023. URL