

**SURFACE DEFECT INSPECTION SYSTEM WITH ANOMALY DETECTION
SYSTEM**

**BY
ANDROW LING KIAN QUAN**

**A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF INFORMATION TECHNOLOGY (HONOURS) COMMUNICATIONS
AND NETWORKING
Faculty of Information and Communication Technology
(Kampar Campus)**

JUNE 2025

COPYRIGHT STATEMENT

© 2025 Androw Ling Kian Quan. All rights reserved.

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Information Technology (Honours) Communications and Networking at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Dr Fityanul Akhyar who has given me this bright opportunity to engage in a deep learning project. It is my first time to investigate the deep learning development as a communication and networking student. Therefore, he gave me much useful knowledge about deep learning and always gave quick response on my questions.

Finally, I would like to thank my parents and my siblings for their love, support, and continuous encouragement throughout my final year project. At the same time, they shared their experiences while they worked on final year project and told me on how to overcome certain problems.

ABSTRACT

This project proposes a **Surface Defect Inspection System with Anomaly Detection Approach** to enhance quality control in manufacturing by leveraging **deep learning** and **computer vision** techniques. The system utilizes the **pretrained ConvNeXt-v2-Tiny** architecture as backbone of model that using **Fully Convolutional Masked Autoencoder** method for self-supervised learning and trained on the **MVTec Anomaly Detection Dataset**, to classify and detect surface defects such as scratches, cracks, and contamination. By employing this strategy, I **fine-tuned** the pretrained model that could adapt to MVTEC Dataset and **extract the good features** from "good" samples only, then evaluate the accuracy on test dataset which includes both "good" and "defective" samples to detect the anomalies on that particular samples by comparing the **Mahalanobis Distance** score with threshold. The final results demonstrate **robust performance** on detecting both "good" and "defective" samples with correct predicted classes and correct predicted defect types on a well-designed web interface. The system aims to address challenges in traditional inspection methods by offering an automated, scalable, and efficient solution for industrial manufacturing sections.

Area of Study: Deep Learning, Computer Vision

Keywords: Robust performance, Limitation, Anomaly Detection, Surface Defect Inspection, pretrained ConvNeXt-v2-Tiny, Fully Convolutional Masked Autoencoder, Fine-tuned, Extract good features, Mahalanobis Distance

TABLE OF CONTENTS

TITLE PAGE	i
COPYRIGHT STATEMENT	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	viii
LIST OF TABLES	ix
LIST OF ABBREVIATIONS	xi
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement and Motivation	1
1.2 Objectives	3
1.3 Project Scope and Direction	3
1.4 Contributions	4
1.5 Report Organization	5
CHAPTER 2 LITERATURE REVIEW	6
2.1 Previous Works of surface defect detection based on Deep Learning	6
2.1.1 Non-contact automated detection	6
2.1.2 Surface Defect Detection Method of Industrial Products Based on Deep Learning	7
2.1.2.1 Supervised Method	7
2.1.2.2 Unsupervised Method	9
2.1.2.3 Weakly Supervised Method	10
2.1.3 Industrial Product Surface Anomaly Detection with Realistic Synthetic Anomalies Based on Defect Map Prediction	11
2.1.3.1 Abnormal Synthesis	11
2.1.3.2 Image Reconstruction Network	14
2.1.4 Image-Based Surface Defect Detection Using Deep Learning	15

2.1.4.1 Image Classification-Based Localization	16
2.1.4.2 Pixel-Based Localization	17
2.1.4.3 Object Detection-Based Localization	17
2.1.5 Hierarchical Image Transformation and Multi-Level Features for Anomaly Defect Detection	18
2.2 Comparison of models' accuracy	21
2.3 Summary	21
CHAPTER 3 SYSTEM METHODOLOGY/APPROACH	22
3.1 System Design Diagram/Equation	22
3.1.1 System Architecture Diagram	23
3.1.2 Use Case Diagram and Description	24
3.1.3 Activity Diagram	25
CHAPTER 4 SYSTEM DESIGN	27
4.1 System Block Diagram	29
4.2 System Components Specifications	29
4.2.1 Hardware	29
4.2.2 Software	29
4.2.3 Model Specifications	29
4.3 Circuits and Components Design	30
4.4 System Components Interaction Operations	31
CHAPTER 5 SYSTEM IMPLEMENTATION	32
5.1 Hardware Setup	32
5.2 Software Setup	32
5.3 Setting and Configuration	32
5.3.1 Model Fine-Tuning	32
5.3.2 Feature Extraction	33
5.3.3 Evaluation	34
5.3.4 Flask Web Interface	35
5.4 System Operation (with Screenshot)	36
5.5 Implementation Issues and Challenges	39

5.6	Concluding Remark	40
CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION		41
6.1	System Testing and Performance Metrics	41
6.2	Testing Setup and Result	42
6.3	Project Challenges	46
6.4	Objectives Evaluation	47
6.5	Concluding Remark	48
CHAPTER 7 CONCLUSION AND RECOMMENDATION		49
7.1	Conclusion	49
7.2	Recommendation	50
REFERENCES		51
POSTER		53

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1.1	Non-contact automated surface detection machine.	7
Figure 2.1.3.1.1	Stages of abnormal synthesis.	11
Figure 2.1.3.1.2	Formula for M_2 .	12
Figure 2.1.3.1.3	Formula for I_a .	12
Figure 2.1.3.1.4	Final synthesized image.	13
Figure 2.1.3.2.1	Formula for L_2 .	14
Figure 2.1.3.2.2	Formula for L_{SSIM} .	14
Figure 2.1.3.2.3	Combined version formula of both loss function.	15
Figure 2.1.4.1.1	Image Classification-Based Localization: Architecture 1.	16
Figure 2.1.4.1.2	Image Classification-Based Localization: Architecture 2	16
Figure 2.1.4.2	Pixel-Based Localization.	17
Figure 2.1.4.3.1	Object Detection-Based Localization: Architecture 1.	18
Figure 2.1.4.3.2	Object Detection-Based Localization: Architecture 2.	18
Figure 2.1.5.1	Hierarchical Image Transformation (HIT).	19
Figure 2.1.5.2	Multi-Level Feature Extraction (MiLF).	19
Figure 2.1.5.3	Combination of HIT and MiLF.	20
Figure 3.1	System Design Diagram	22
Figure 3.1.1	System Architecture Diagram	23
Figure 3.1.2	Use Case Diagram	24
Figure 3.1.3	Activity Diagram	25
Figure 4.1	System Block Diagram	27
Figure 4.3	Circuits and Components Design Layer	30
Figure 5.3.1.1	Pretrained Model Retrieved	33
Figure 5.3.1.2	Fine-tuned model checkpoint	33
Figure 5.3.2.1	Good Features Saved File	33
Figure 5.3.2.2	Feature Vectors Extracted	34
Figure 5.3.3.1	Threshold Saved File	34
Figure 5.3.3.2	Example Evaluation Performance Metrics	35
Figure 5.3.4.1	Folder Upload & Extension Images	35

Figure 5.3.4.2	Local Hosting	35
Figure 5.3.4.3	Front End Index Template	35
Figure 5.4.1	Product Category List	36
Figure 5.4.2	Select Product Category	36
Figure 5.4.3	Upload Image from Same Category	37
Figure 5.4.4	Product Category & Image Uploaded	37
Figure 5.4.5	Analysis Result	38
Figure 6.1.1	Performance Metrics	41
Figure 6.1.2	Data Labeling	41
Figure 6.2.1	Bottle Matrix	42
Figure 6.2.2	Cable Matrix	42
Figure 6.2.3	Capsule Matrix	42
Figure 6.2.4	Carpet Matrix	42
Figure 6.2.5	Grid Matrix	42
Figure 6.2.6	Hazelnut Matrix	43
Figure 6.2.7	Leather Matrix	43
Figure 6.2.8	Metal Nut Matrix	43
Figure 6.2.9	Pill Matrix	43
Figure 6.2.10	Screw Matrix	43
Figure 6.2.11	Tile Matrix	43
Figure 6.2.12	Toothbrush Matrix	43
Figure 6.2.13	Transistor Matrix	44
Figure 6.2.14	Wood Matrix	44
Figure 6.2.15	Zipper Matrix	44
Figure 6.2.16	Web Interface Result	45

LIST OF TABLES

Table Number	Title	Page
Table 2.2	Comparison of models' accuracy	21
Table 4.2.1	Hardware Components	29
Table 4.2.2	Software Components	29
Table 4.2.3	Model Specifications	29
Table 5.1	Hardware Setup	32
Table 5.2	Software Setup	32
Table 6.2	Evaluation Results	42

LIST OF ABBREVIATIONS

<i>FCN</i>	Fully Convolutional Network
<i>PCB</i>	Printed Circuit Board
<i>RPN</i>	Region Proposal Network
<i>Faster R-CNN</i>	Faster Region Convolutional Neural Network
<i>R-CNN</i>	Region Convolutional Neural Network
<i>Mask R-CNN</i>	Mask Region Convolutional Neural Network
<i>DBN</i>	Deep Belief Network
<i>SOM</i>	Self-Organizing Map
<i>GAN</i>	Generative Adversarial Network
<i>RBM</i>	Restricted Boltzmann Machines
<i>DS</i>	Differential Search
<i>TSV</i>	Through-Silicon Via
<i>HIT</i>	Hierarchical Image Transformation
<i>MiLF</i>	Multi-Level Features Extraction
<i>RAM</i>	Random Access Memory
<i>FCMAE</i>	Fully Convolutional Masked AutoEncoder

Chapter 1

Introduction

In this modern era, it is crucial for manufacturers to maintain high quality standard of their product so that they will gain competitive advantage compared to other manufacturers. Surface defect is when an original object image discovered any irregularity on the surface of particular object such as scratch, crack, corrosion and more. Therefore, when the products produced by manufacturers involve surface defects, it can significantly affect the functionality and structure of the product which could potentially harm the public. Besides that, traditional inspection methods are more rely on physical observation and may fail to identify some unexpected defects or anomalies then lead to quality issues [1].

Therefore, developing an automated Surface Defect Inspection System with Anomaly Detection Approach offers guaranteed solutions towards the manufacturing process while the Anomaly Detection is the process of identifying samples that are out of the ordinary in relation to regular patterns in dataset when the only available training data is defect-free images [2]. This system involves advanced computer vision and deep learning techniques to do the automate detection and classification of surface defects in real-time. Besides that, this approach enhances the accuracy and efficiency of surface inspections by reducing human work and error, which ensures most of the surface defects are captured and solved.

1.1 Problem Statement and Motivation

Problem Statement

i. Challenges in efficiency and accuracy during manufacturing process

The manufacturers must ensure their products are in good condition to maintain the competitive advantage among the competitors and fulfil customer satisfaction. However, using traditional inspection system still consists of unforeseen defects which could lead to quality failures as the manufacturing processes are becoming more complex and relieved heavily on human inspection. When it relies heavily on human inspection, the risk of human error occurring will rapidly increase and the time-consuming on human inspection is longer as they need to observe the surface defects manually which could reduce their production efficiency. Therefore, an automated inspection system with computer vision and deep learning techniques that enhance efficiency and accuracy at the same time can address these issues.

ii. Challenges in detecting new types of surface defects

As the advancement of modern manufacturing, the production process continues to evolve and bring various types of new surface defects that are not found in existing inspection systems. Additionally, traditional inspection system mostly was based on predefined defect libraries which means that whenever there is any new surface defect that is not included in the defect libraries, the system unable to detect it and leads to unexpected functionality problems then product quality is not guaranteed. To overcome this problem, a surface defect inspection system that uses computer vision and deep learning techniques can dynamically detect both known and unknown defects and anomalies.

iii. Challenges in real-time continuous quality monitoring

In this current fast-paced manufacturing process, the ability to detect surface defects in real-time is much better compared to traditional inspection systems. For traditional inspection system, it is much slower and less efficient which causes unable to catch up with the modern production lines and leads to slow productivity of products. Therefore, a real-time surface defect inspection system that applied anomaly detection is needed to identify and classify both expected and unexpected surface defects, continue to monitor product quality and reduce the risk of defective products reaching to the market.

Motivation

The aim of the thesis is to propose a new surface defect inspection system with anomaly detection approach. The system will implement techniques of computer vision and deep learning. The primary motivation is to detect the defects and anomalies on particular object such as scratch, cracks, corrosion and more in fast and accurate situations. Therefore, this system can be used effectively in manufacturing sector because they need to ensure that their products don't have any defects on it only can be released to the market for sales. The secondary motivation is to help out the manufacturer to accomplish their works in a short time as the system is in automated pattern with accurate detection on defects towards the objects.

1.2 Objectives

The aim of the project is to **develop a deep learning model** that has the function of surface defect inspection system with anomaly detection approach. The model is trained by using combined multi-category dataset from MVTec Anomaly Detection Dataset [3]. Next, **implement and fine-tune a ConvNeXt-based convolutional neural network (CNN)** to classify the good and defective samples among different product categories. Moreover, **extract the good features from “good” samples** to allow the fine-tuned model to differentiate the good and defective parts effectively and **evaluate the model’s performance** by comparing the Mahalanobis Distance score with the threshold for each product category. Additionally, deploy the model onto a well-designed web interface for anomaly detection.

1.3 Project Scope and Direction

This project focuses on the development of an automatic surface defect inspection system utilizing an anomaly detection approach. The system is designed to identify surface anomalies without the need for explicitly labeled defective samples during training. The scope of the project includes:

- Training deep learning models (specifically ConvNeXt) on “good” (non-defective) and “defective” samples from the MVTec dataset.
- Using a deep learning framework (PyTorch) to fine-tune and implement the ConvNeXt model.
- Extract the good features from “good” samples by using the fine-tuned model.
- Evaluating the model trained performance on multiple categories within the MVTec dataset, such as bottle, cable, capsule and others by comparing the Mahalanobis Distance score with threshold.
- Deploy the model onto well-designed web interface to allow the user to choose the image and determine whether the image is predicted as “good” or “defective”.

Direction:

The project progresses through a series of structured stages:

1. Literature Review:

Studying state-of-the-art methods for anomaly detection and surface defect inspection system using deep learning.

2. **Model Development:**

Utilizing a ConvNeXt model with pretrained ImageNet weights as the base model. Fine-tuning it for one-class anomaly detection.

3. **Training Strategy:**

Training models for each category and saving them into single pth file to capture good and defective features from category samples.

4. **Anomaly Detection and Evaluation:**

Testing the models to identify anomalies and evaluate their performance using metrics such as graph of accuracy and confusion matrix.

5. **Result Analysis and Discussion:**

Analyzing the detection performance on web interface, identifying the problems and proposing improvements.

1.4 Contributions

My surface defects inspection system with anomaly detection approach used computer vision and deep learning techniques is developed by **integrating the finetuned ConvNeXt v2 with FCMAE self-supervised learning combined with Mahalanobis Distance** to define the anomalies [4]. This combination shows that strong pretrained representations while keeping the detection system **lightweight and efficient**. Besides that, by **computing the per-category mean and covariance statistics** for good samples, the inspection system is allowed to **adapt threshold for each product category** which is very important factors because different categories always have their own different characteristics. Moreover, the system **does not rely on the global threshold** but **computing the threshold from validation tuning** and loading it to **ensure that each product category's boundary is optimized**. Additionally, my inspection system is **deployed by using the lightweight deployment method such as Flask**, unlike the other inspection system that only publish results in paper. My system **allows the real-time user to do their product anomaly detection** by uploading the images, categories selection and get the results automatically, which is **crucial to current fast-paced manufacturing environments** as the system **can reduce human work** for more complex tasks. In the end, the **product quality can be improved** because the system enhanced the overall process for **quality assurance with consistent performance** and **prevented the defective products from being leaked into the market**.

1.5 Report Organization

This report is organized into 6 chapters: Chapter 1 Introduction, Chapter 2 Literature Review, Chapter 3 System Design, Chapter 4 System Implementation and Testing, Chapter 5 System Outcome and Discussion, Chapter 6 Conclusion. The first chapter is the introduction of this project which includes problem statement, project background and motivation, project scope, project objectives, project contribution, and report organization. The second chapter is the literature review carried out on several existing surface defect inspection systems in the market to evaluate the strengths and weaknesses of each system. The third chapter discusses the overall system design of my inspection system. The fourth chapter is about the details on how to implement the design of the system. Furthermore, the fifth chapter reports the results of my inspection system on how the system detects image as normal or defective follow with the anomaly score and the predicted defect types. The last chapter is about the conclusion of my overall project which includes the important elements of my inspection system that make my system works well.

Chapter 2

Literature Review

2.1 Previous Works of surface defect detection based on Deep Learning

2.1.1 Non-contact automated detection

A non-contact automated detection is a surface defect detection that is based on visual perception technology. In current manufacturing environment, it can operate well for a long period with high accuracy and efficiency. Deep learning-based surface defect detection uses automated feature extraction techniques to perform but it requires large training data. By using deep learning models, it can detect defects based on dataset and transferred to achieve high success rate. Additionally, these models give adaptable algorithms to detect various defects without any code needed for each defect [5].

In order to perform automated surface defect detection, a computer vision system is needed to utilize cameras and sensors to capture high-resolution images for analysis. Computer vision is a part of artificial intelligence which includes methods like acquiring, processing, analyzing, understanding digital images and extraction of high-dimensional data. Therefore, the system process will involve image preprocessing, extraction of features, and analysis using deep learning models on labelled datasets.

The features extraction is important because it could identify the image characteristics for defect classification. In recent years, the features extraction process is transformed to automatic process to do the extraction from raw images and integrated into neural network architecture.

Strengths:

- The surface defect inspection system applied with advanced algorithms and deep learning models to enhance the accuracy of defect detection.
- Applied with the computer vision to perform automated surface detection process by using features extraction in real-time to classify the defects, improved the efficiency of manufacturing process instead of using manual inspection.
- A friendly system to apply on majority of industries and compatible with large-scale industrial because of its adaptability.

Limitations:

- Automated surface defect detection required large amounts of high-quality datasets to do model training.
- When all techniques and models implemented into the surface defect inspection system are complex which required the people who experienced with it to complete the setup.

Non-contact automated surface detection



Figure 2.1.1 Non-contact automated surface detection machine

2.1.2 Surface Defect Detection Method of Industrial Products Based on Deep Learning

2.1.2.1 Supervised Method

The supervised method needs the labelled training set and the unlabelled test set as to find the inherent laws of samples then apply it on the test set. Next, supervised method can be separated into metric learning based and representation learning based. There are many models in this domain such as Siamese Network, ShuffleNet, Fully Convolutional Networks (FCN), Mask R-CNN and more [6].

Firstly, Siamese Network is functioned to measure the similarities between two types of samples by minimizing the distance of similar categories and maximizing the distance of

different categories. This technique had been effectively used in detecting defects on printed circuit boards (PCBs) and detecting cross-class defects in real-world factory datasets.

Secondly, ShuffleNet is a lightweight and efficient type of network using pointwise group convolution and channel shuffle to maintain accuracy while reducing computational costs. Nowadays, some of the industrial applied this network on their industrial inspection systems to do defect detection in plastic containers and surface defect detection with high accuracy on datasets like NEU.

Thirdly, Faster R-CNN is an advanced version that is built based on Fast R-CNN by implementing a Region Proposal Network (RPN) to improve the speed and efficiency of defect detection. This technique is used to detect defects of power line insulator, PCB surface, and others that require accurate localization and classification of defects.

Moreover, Fully Convolutional Networks (FCN) is an image segmentation approach where all layers of image are convolutional and enable pixel-level prediction with precise segmentation. Therefore, FCN mostly was used for defect detection in various contexts which include keyboard light leakage, insulator string and solar cell images, then combined with techniques like U-net architecture.

Last but not least, Mask R-CNN is an extension of Faster R-CNN which combined object detection with instance segmentation to allow detection and masking of defects in two stages. It is very effective on performing high-accuracy tasks like solar panel pollution detection or tunnel defect detection for further improving feature extraction and defect masking.

Strengths:

- Supervised method able to achieve very high accuracy in defect detection due to the precise training on labelled datasets if the datasets is sufficient.
- Well-established models like Siamese Networks, ShuffleNet, Faster R-CNN, FCN and Mask R-CNN are widely used in overall industrial applications by providing reliable and robust defect detection
- Techniques like Mask R-CNN have advanced features like precise localization and masking but not only for defect detection.

Limitations:

- Models require significant computational resources which make them not suitable for real-time detection.
- There will be a risk of overfitting when the labelled dataset is too small and leads to poor generalization on new data.
- Need a large and well-labelled dataset which is expensive and time-consuming.

2.1.2.2 Unsupervised Method

In unsupervised learning, the machines learn patterns from unlabeled data and classify it to new data based on learned models which overcome the limitations supervised method like the need for labeled data. The most common unsupervised methods are reconstruction-based methods and embedding similarity-based methods. Additionally, Deep Belief Networks (DBN) and Self-Organizing Maps (SOM) are also utilized [5].

Autoencoder is one of the reconstruction-based methods. The goal of autoencoder is to learn and reconstruct the input signal, with anomalies detected through poor reconstruction because autoencoder consists of an encoder to learn low-dimensional features and decoder to reconstruct the input signal. Next, Generative Adversarial Network (GAN) is also a reconstruction-based method. It involves a generator for learning data distribution and discriminator for estimating the probability of data being real or fake, performing the anomalies detection by generating and comparing defect likelihood maps. Additionally, recent advances such as Masked Autoencoders extend this concept by reconstructing missing image patches, allow model to learn robust feature representation in self-supervised method [4].

Furthermore, Deep Belief Network (DBN) composed of multiple layers of Restricted Boltzmann Machines (RBMs) and were trained layer by layer to map training samples to non-defect images. This network was used in defect detection for solar cells and bolt defect identification, often optimized with algorithms like Differential Search (DS) for better performance.

Other than that, Self-Organizing Map (SOM) mimics the brain's neural network, classifying input patterns by finding optimal reference vectors. SOMs are useful for distinguishing normal and defective areas. This method was applied in detecting wood defects and TSV (Through-

Silicon Via) defects, with combined techniques like Otsu for enhanced detection and localization.

Strengths:

- Able to learn patterns from unlabelled data, solve the limitations of needing large, labelled datasets.
- Effective on identifying anomalies by highlighting deviations from normal patterns.
- It can adapt more easily to new data because they did not bound by specific labels.

Limitations:

- The accuracy of unsupervised method is low especially for those complex defects.
- Difficult in detecting all types of defects especially for those defects that do not appear during training.

2.1.2.3 Weakly Supervised Method

Weakly supervised method combined the ability of both supervised and unsupervised learning while minimizing the costs associated with extensive labelling. Currently, the commonly used weakly supervised method is incomplete supervision method and inexact supervision method [5].

Incomplete supervision method has the most training samples unlabeled and a few labelled samples which are unable to train a robust model. Among the incomplete supervision methods, the semi-supervised approach is commonly used for detecting defects of industrial surface as it can develop unlabeled sample data automatically without any manual intervention. Next, inexact supervision deals with imprecise monitoring information, where labels are coarse-grained such as image-level labels instead of pixel-level labels.

Strengths:

- Reduce the need for extensive labelling.
- Able to achieve better performance compared to unsupervised methods even though there is only a limited amount of labelled data.
- More flexible, can be applied in many situations because only they need coarse-grained labels.

Limitations:

- The result is inconsistent because it more depends on quality and quantity of labelled data available.
- It is more complex compared to supervised or unsupervised methods.
- Limited functionality towards current industrial applications.

2.1.3 Industrial Product Surface Anomaly Detection with Realistic Synthetic Anomalies Based on Defect Map Prediction

2.1.3.1 Abnormal Synthesis

Defects that can be commonly understood as the situation where the contextual information of a certain region on the foreground target is significantly different from the surrounding areas and not related to the target background. It emphasizes the authenticity of synthesizing anomalies and can be divided into three stages on process of generating synthetic abnormal images [7].

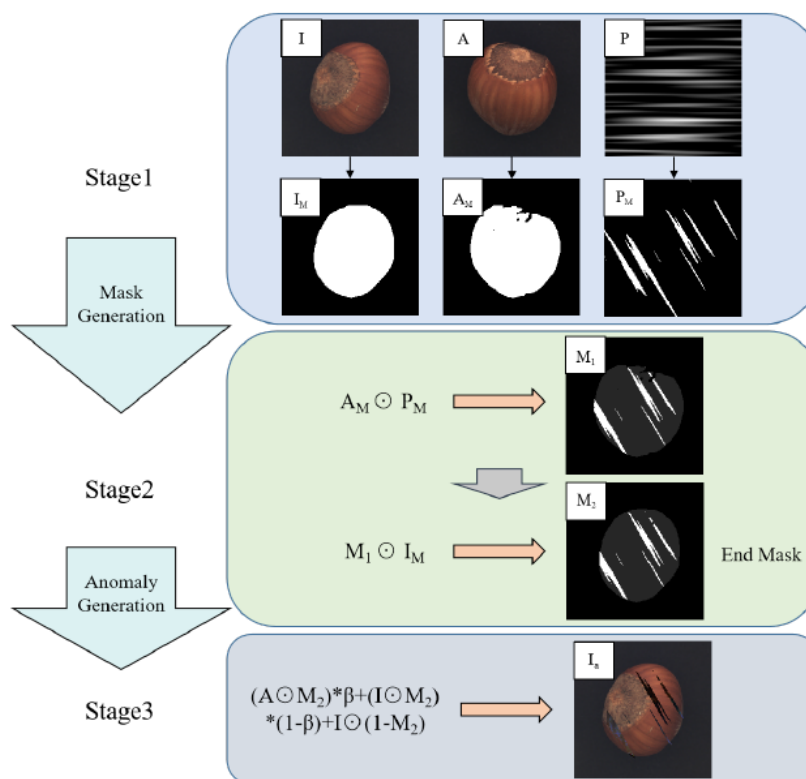


Figure 2.1.3.1.1 Stages of abnormal synthesis

In first stage, select an input image I and choose a random image A from dataset to serve as the source of anomalies. Next, the foreground object from I and A will be obtained by using edge detection or grayscale thresholding and results mask images I_M and A_M . Then, use Perlin noise generator to create random noise texture image P which threshold to produce binary mask image P_M .

In second stage, the synthetic anomaly must appear on the foreground object. Therefore, the image A_M will first multiplied pixel-wise with P_M to get the mask image M_I . Then, the image M_I is multiplied pixel-wise with I_M to obtain final mask image M_2 .

$$M_2 = A_M \odot P_M \odot I_M$$

Figure 2.1.3.1.2 Formula for M_2

In third stage, M_2 is used to extract corresponding regions from sample image A and input image I , then use random interpolation to obtain the final defect image. The final image composition will be the defect region combined with other parts of input image I that unaffected by mask and creating a final synthesized image I_a .

$$I_a = (A \odot M_2)\beta + (I \odot M_2)(1 - \beta) + I \odot (1 - M_2)$$

Figure 2.1.3.1.3 Formula for I_a

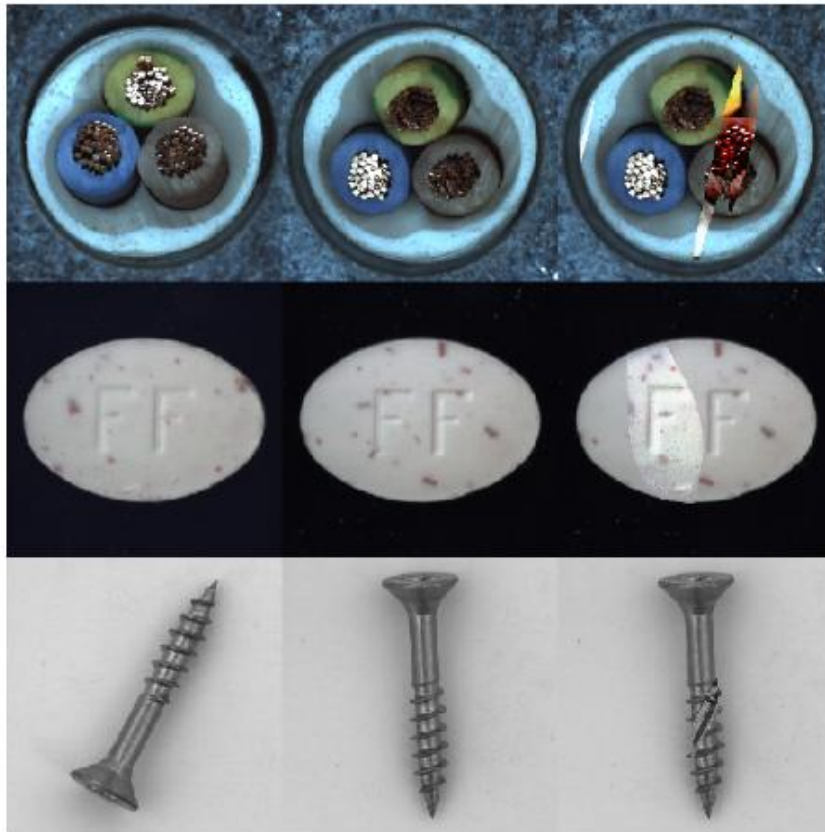


Figure 2.1.3.1.4 Final synthesized image

Strengths:

- Generate realistic anomalies that mimic real-world defects which enhance the model to detect actual defects.
- Able to create various defect types by using random images from dataset as source of anomalies.
- Save time and effort in anomaly detection where able to create synthetic defect image without manual labelling.

Limitations:

- Since the method relies on Perlin noise and random interpolation, it not realistic for some special cases like in complex industrial environments.
- Synthetics anomalies could lead to overfitting if models become too specialized on generated anomalies.
- Multi-stage processes are expensive which could slow down the model training on large datasets.

2.1.3.2 Image Reconstruction Network

Autoencoder is widely used in anomaly detection and some certain image reconstruction tasks. It consists of encoder and decoder which function to compress the input into a lower-dimensional representation and reconstructs the original image from the compressed data respectively.

The deep feature extractor corresponds to the encoder part of the autoencoder, but it does not involve any network parameter updates. Deep feature extractor is used to capture high-level semantic information and the parameters from encoder will be copied to feature extractor to ensure that the reconstructed images are similar with the original image in both visual and terms of feature representations.

Furthermore, the L_2 loss function is a common method to compute the pixel-wise differences between generated and real images. However, L_2 loss function is sensitive to noise and could perform poorly on image edges.

$$L_2(I, I_r) = \frac{1}{HW} \sum_{i=1}^H \sum_{j=1}^W \|I_a(i, j) - I(i, j)\|^2$$

Figure 2.1.3.2.1 Formula for L_2

Moreover, the L_{ssim} loss function is used to measure the structural similarity between generated image and original image, compensating for the failure comes from L_2 loss function.

$$L_{SSIM}(I, I_r) = \frac{1}{HW} \sum_{i=1}^H \sum_{j=1}^W 1 - SSIM(I, I_r)_{(i,j)}$$

Figure 2.1.3.2.2 Formula for L_{SSIM}

In order to enhance the reconstruction of image, both loss functions are combined to form a visual image reconstruction loss function that is used to compute the loss of image reconstruction in terms of visual perception. A deep feature loss also incorporated to ensure that the generated image retains high-level semantic similarity with the original.

$$L_{rec}(I, I_r) = \lambda_2 L_{vision}(I, I_r) + \lambda_3 L_{deep}(z, \hat{z})$$

Figure 2.1.3.2.3 Combined version formula of both loss functions

Strengths:

- Compress the input data into lower-dimensional representations which helps in capturing essential features and reduce noise.
- Use deep feature extractor to detect subtle and complex anomalies in images.
- More robust to noise and better at preserving semantic features in images by using multiple loss functions.

Limitations:

- L_2 loss function sensitive in noise which could degrade the quality of reconstructions.
- L_2 loss function performs poorly in image edges.
- Lack of parameter update in feature extractor which could limit the effectiveness of model in detecting anomalies.

2.1.4 Image-Based Surface Defect Detection Using Deep Learning

Surface defect detection is vital in manufacturing as it is used to ensure quality control and reduce wastage by identifying defects early in the production line [7]. The industry often uses image processing techniques on defect detection, but they are always facing the problem of handling noise, lighting variations and complex textures. Therefore, in recent years, deep learning has been transformed into a more robust method for defect detection due to its flexibility and adaptability for different types of defects. [8]

Deep learning excels in identifying complex surface inspection as it has the ability to learn the features automatically from datasets and make it more adaptable to different kinds of defects. Although traditional techniques such as edge detection and image segmentation worked well for consistent surface defects, they lack the ability to clarify the defects in shape, size or intensity.

There are several architectures categories for defect localization and classification which includes image classification-based localization, pixel-based localization and object detection-based localization.

2.1.4.1 Image Classification-Based Localization

There will be two types of architecture for image classification. First architecture uses the entire image for input and only output the image which got defect. However, the defect's location is not specified so there is a possibility where the whole image can be considered as defect. Therefore, this architecture is commonly known as image classification, and it is a simple approach which can miss some small defects from poor localization accuracy.

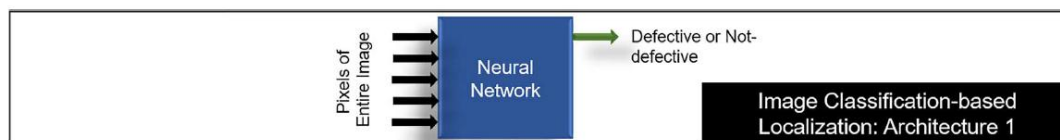


Figure 2.1.4.1.1 Image Classification-Based Localization: Architecture 1

For second architecture, it uses preprocessing units to perform segmentation on entire image into small image called patch. Most common preprocessing unit will be using sliding window to divide the image into patch then will pass into neural network and classified it with defect types.

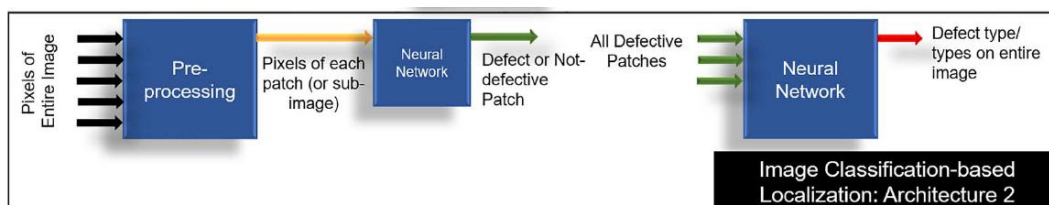


Figure 2.1.4.1.2 Image Classification-Based Localization: Architecture 2

Strengths:

- Straightforward image classification architecture, less computational power.
- Effective in identifying large and obvious defects across entire image.
- Implement image segmentation in second architecture to enhance the accuracy on detecting smaller defects.

Limitations:

- Poor localization in first architecture as it cannot locate the specific defects' location.
- It has high possibility of missing small or subtle defects.
- High cost in using sliding window on second architecture and time-consuming.

2.1.4.2 Pixel-Based Localization

This architecture performs at the pixel level which means that input will be the entire original image, and output will be the image with probabilities of defect on each pixel. Unlike image classification-based localization, this architecture able to locate the defect on the image accurately.

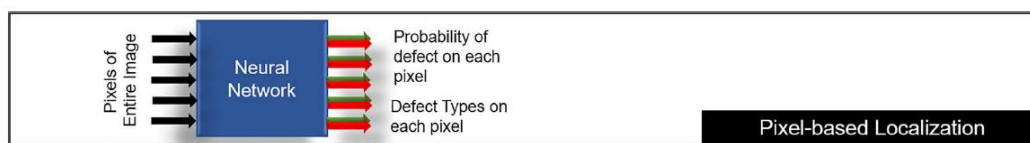


Figure 2.1.4.2 Pixel-Based Localization

Strengths:

- Provide precise localization of defects in pixel level.
- Includes a probability map that highlights the likelihood of defects at each pixel.
- Ideal for complex features as it inspects every pixel to ensure the smallest defects are detected as well.

Limitations:

- Sensitive to noise in image which could lead to false positives.
- High computational requirement as it is specific in pixel level to do detection.
- More complex compared to other architectures as requiring more sophisticated neural network architecture.

2.1.4.3 Object Detection-Based Localization

There are two types of architectures within the object detection-based localization. For architecture one, it treats defects as an object and uses region proposal network (R-CNN) to localize defects with bounding boxes.

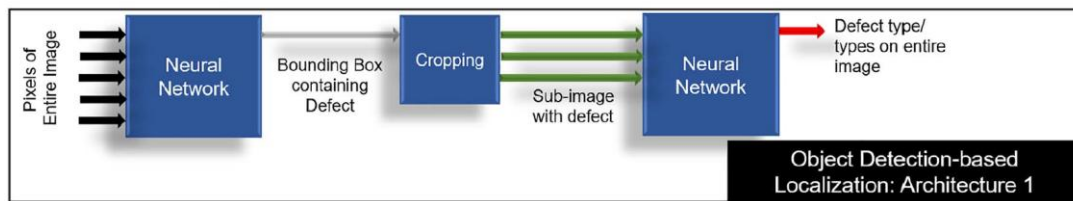


Figure 2.1.4.3.1 Object Detection-Based Localization: Architecture 1

For second architecture is an extension of object detection-based localization which can perform localization and defect classification. Normally, this architecture will be implemented with several fully connected layers at the end of faster-RCNN to do defects classification.

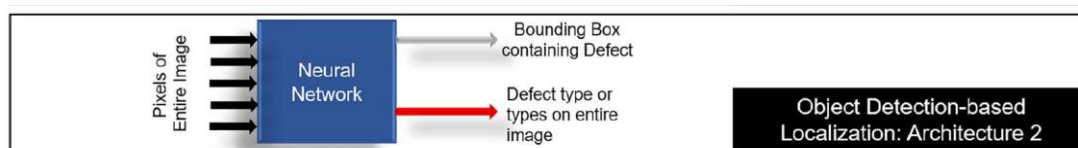


Figure 2.1.4.3.2 Object Detection-Based Localization: Architecture 2

Strengths:

- Standardized framework in object detection, making it easier to implement into existing system for industrial defect detection.
- Good in detecting wide variety of defects.
- A more complete version of object defect detection system as both defect detection and localization are performed well.

Limitations:

- Struggle dealing with multiple small defects in close proximity.
- Require high data for both defect classification and localization which may not always be available in real-world industrial settings.
- Struggle dealing with edges cases.

2.1.5 Hierarchical Image Transformation and Multi-Level Features for Anomaly Defect Detection

This article proposes a Hierarchical Image Transformation (HIT) approach which is used to generate new transformation of training datasets that simulate real-world noise and perturbations [9]. This approach apparently benefits the environment but acquiring ideal

samples is challenging. However, introducing noise into the training process will make the model become more resilient to non-ideal testing conditions. Within the transformations include Hue saturation adjustments, noise injection, shadow effect addition and brightness and contract adjustment. This approach helps the model to learn a wide range of feature variations and results in it becoming more robust in detecting anomalies in non-ideal images.

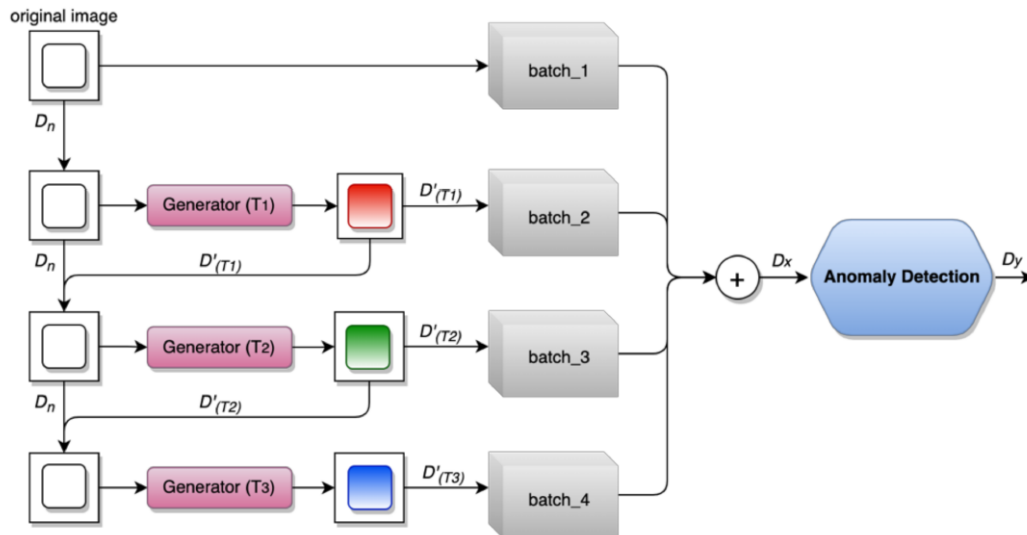


Figure 2.1.5.1 Hierarchical Image Transformation (HIT)

Other than that, this article also proposes Multi-Level Feature Extraction (MiLF) [9]. Basically, it is an enhanced version in capturing features at different layers of CNN. In this article it mentioned that ResNet architecture has multiple layers of level abstraction which include low-level features, middle-level features and high-level features. Each of it handles different level features like low-level features focus on pixel information, middle-level features focus on capturing more abstract patterns and high-level features will focus on object-level information.

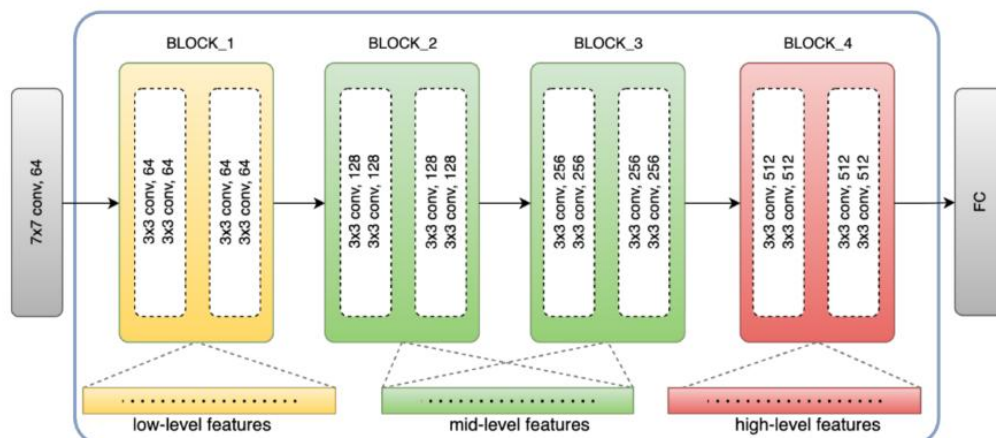


Figure 2.1.5.2 Multi-Level Feature Extraction (MiLF)

By combining features from different levels, the model able to reach a higher accuracy on detecting normal and anomalies images.

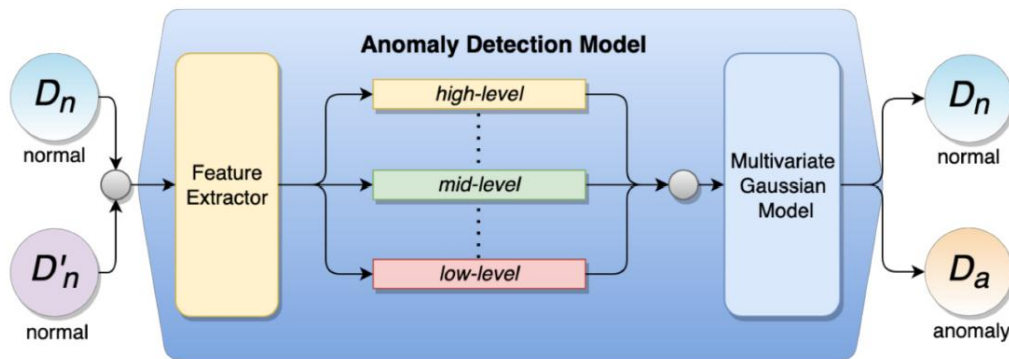


Figure 2.1.5.3 Combination of HIT and MiLF

Strengths:

- HIT approach enhances the model's resilience to noise by simulating real-world disturbances which can handle various real-world scenarios.
- HIT able to learn the features in wide range of feature variation.
- More flexible in detecting subtle and complex anomalies by involving the combination of low, middle and high-level features.

Limitations:

- Combination of HIT and MiLF increase the complexity of model design.
- High inference time due to the complexity of extracting and processing features from multiple layers.

2.2 Comparison of Models' Accuracy

Models	Types	Pretrain	ImageNet-1K (Top 1%)	ImageNet-1K (Top 5%)	Size
ConvNeXt	Convolutional network	ImageNet- 21K	82.90	96.62	Tiny
			84.59	97.41	Small
			85.81	97.86	Base
			86.61	98.04	Large
			86.97	98.20	xLarge
Swin- Transformer V2	Transformer	ImageNet- 21K	86.17	97.88	Base
			86.93	98.06	Large
DeiT-3	Transformer	ImageNet- 21K	83.06	96.77	Small
			84.56	97.19	Medium
			85.70	97.75	Base
			86.97	98.24	Large
			87.19	98.26	Huge
EfficientNet V2	Convolutional network	ImageNet- 21K	84.29	97.26	Small
			85.47	97.76	Medium
			86.31	97.99	Large
			86.39	97.83	xLarge
RepLKNet	ConvNeXt + Large kernels	ImageNet- 21K	85.20	97.56	Base
			86.63	98.00	Large

Table 2.2 Comparison of Models' Accuracy [10]

2.3 Summary

This literature review emphasizes various methods and techniques that can be used for surface defect inspection in industrial settings, focusing on the importance of deep learning and computer vision applied in surface defect detection. By exploring strengths and limitations of all techniques and approaches, result a complete approach can be applied in industrial settings with realistic ways.

Chapter 3

System Methodology/Approach

3.1 System Design Diagram/Equation

In this diagram shows the **overall process** on how the inspection system with anomaly detection approach being developed against MVTec dataset which include fine-tuning train phase, feature extraction phase, anomaly detection phase and web interface phase.

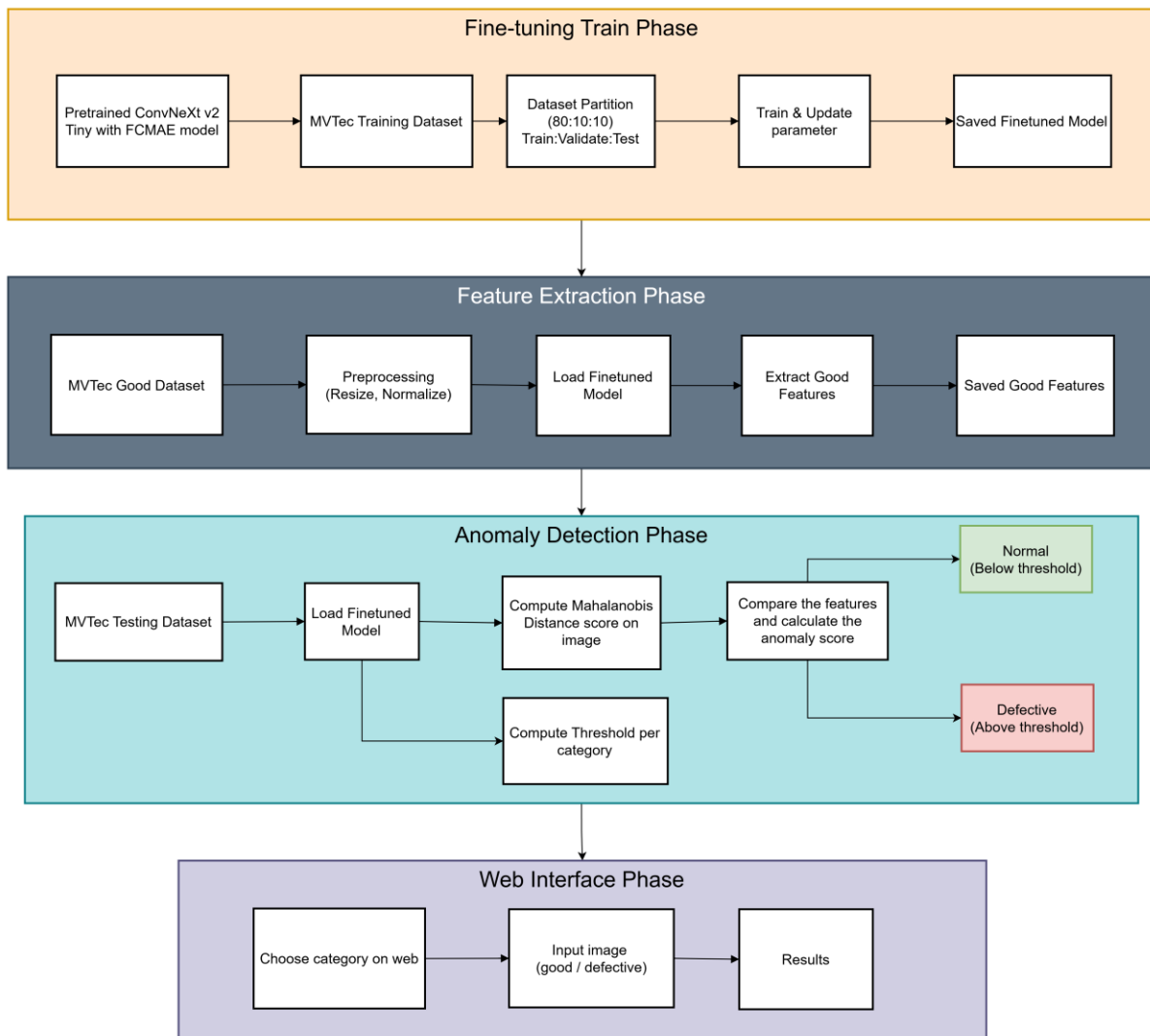


Figure 3.1 System Design Diagram

3.1.1 System Architecture Diagram

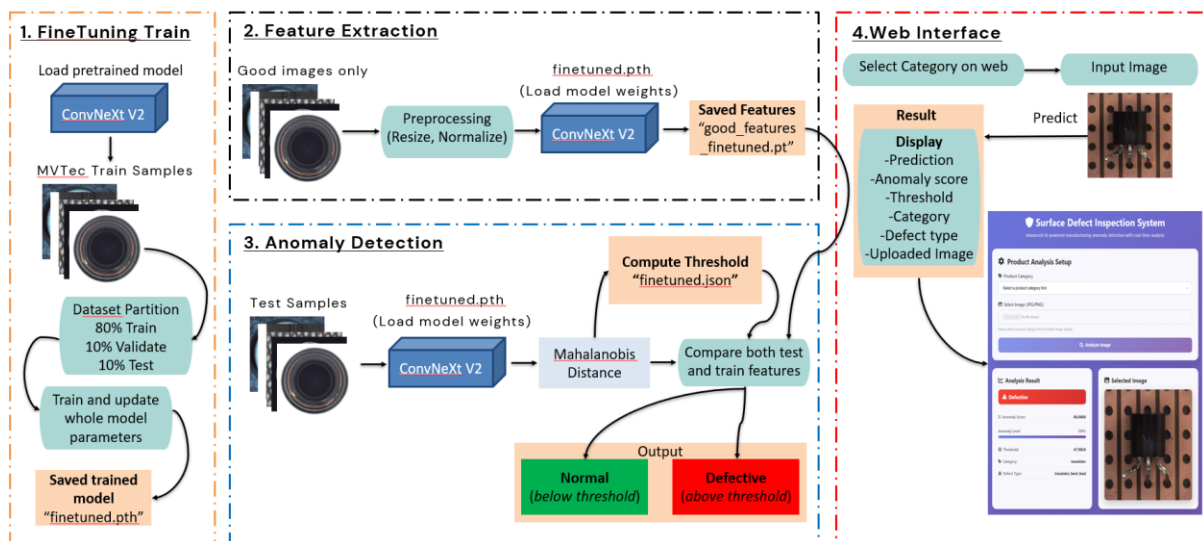


Figure 3.1.1 System Architecture Diagram

The system architecture diagram of my surface defect inspection system is designed by including fine-tuning process, feature extraction process, anomaly detection which is also evaluation process and also the web interface process.

During the fine-tuning process, I used the pretrained ConvNeXt v2 model to train on the MVtec training dataset which includes both “good” and “defective” images. Before the fine-tuning process, the dataset will be separated as 80%, 10% and 10% for the use of training, validation and testing respectively. Then the training process will occur and loop to update the pretrained model parameters to be adapted on MVtec dataset. Next, save it into a model checkpoint called “finetuned.pth”.

For the feature extraction process, I load the “good” images first and do the image preprocessing such as resize and normalize. Once the preprocessing had been done, load the finetuned model to extract the good features from it and saved it as “good_features_finnetuned.pt”.

The third phase which is anomaly detection phase, load the testing dataset which includes both “good” and “defective” images as well. Next, use the finetuned model to compute the Mahalanobis Distance between image’s features vector and stored distribution of good samples [11]. At the same time, compute the threshold per category and saved it as “finetuned.json” so

that the result could be optimized and efficient. Once compute the distance and threshold, compare it to get the results by looking at the anomaly score whether it is below or above the threshold. If the anomaly score is below threshold, the image will be predicted as “normal”. In the other hand, the image will be predicted as “defective” if it is above the threshold.

Lastly, the inspection system is deployed onto the web by using lightweight deployment, Flask. On the web, the real-time users are required to select the product category before uploading the image. Once the product category is selected and the image is uploaded, analyze it then the analysis result will come out which includes the prediction results, anomaly score, threshold used, category, predicted defect type and uploaded image.

3.1.2 Use Case Diagram and Description

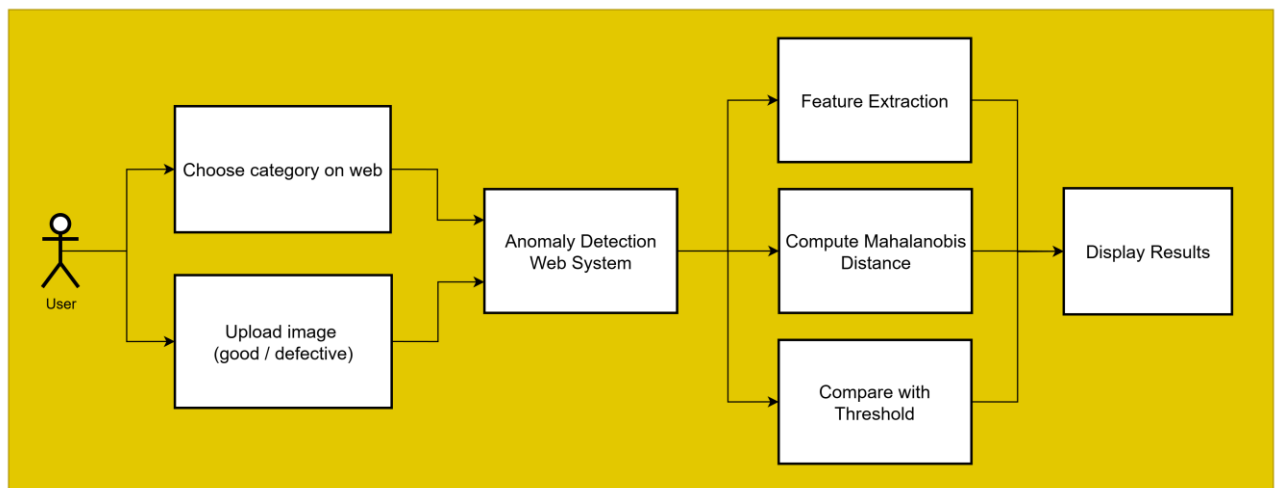


Figure 3.1.2 Use Case Diagram

Choose category

- Allow the user to choose categories of MVTec dataset such as bottle, capsule, cable, transistor and more.
- It is a required option that needs to be fulfilled before uploading the image.

Upload Image (good/defective)

- Allow the user to upload the image of MVTec dataset no matter if it is good images or defective images.

Anomaly Detection Web System

- The overall backend system that used to perform anomaly detection on particular images.

Feature Extraction

- Important steps to perform in order to compare the images feature mean vector.

Compute Mahalanobis Distance

- Calculate the distance score on that particular image to determine its anomaly score.

Compare with Threshold

- Use anomaly score compared with the threshold to decide whether the image is normal or defective in result.

Display results

- The final part shows the results which include image prediction result, anomaly score, threshold used, product category, predicted defect type and the uploaded image.

3.1.3 Activity Diagram

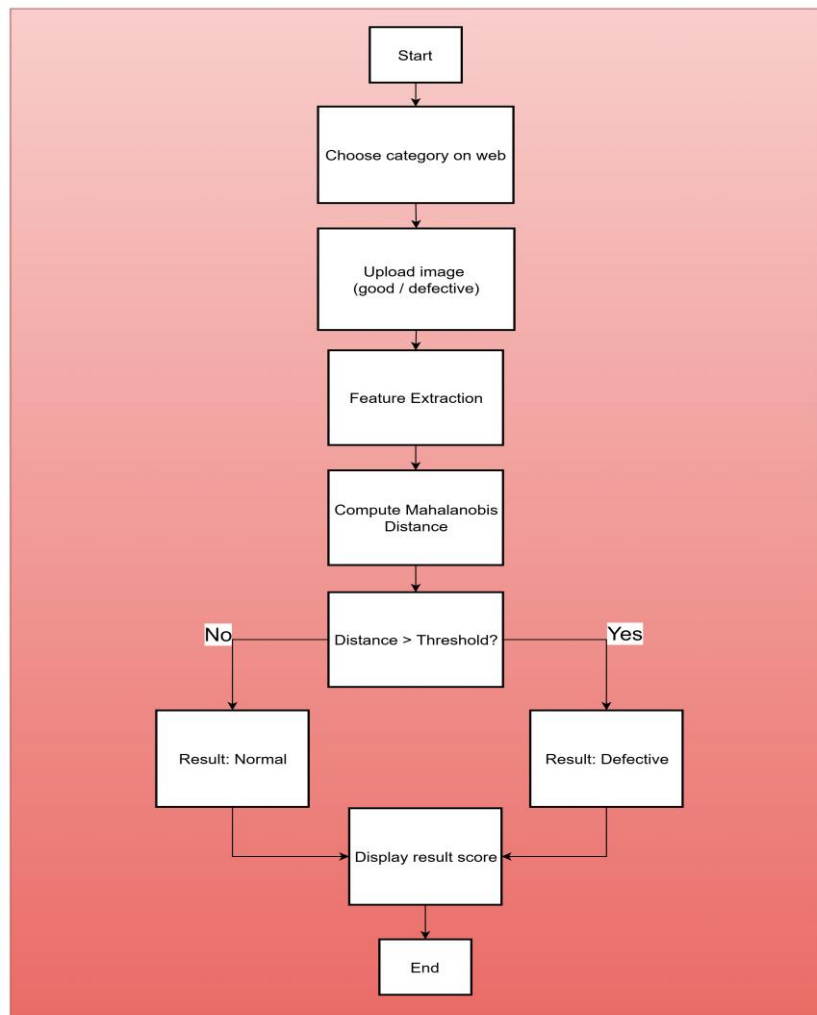


Figure 3.1.3 Activity Diagram

In this diagram shows the workflow starting of the inspection system until the end of the system. In the beginning, the user will enter to the web interface of my inspection system and select category from the dropdown options so that the user can upload the image as well. Next is the process of feature extraction where the ConvNeXt v2 will be responsible to extract the features of that uploaded image to do comparison with the good features of image. Then the system will start to compute the Mahalanobis distance score to see whether it is larger or smaller than the precalculated threshold. If the distance is smaller than threshold, which means that the image is normal image. In the other hand, if the distance is larger than threshold, which means that the image is defective. After the model decide the image result, it will display out the result score such as prediction result, anomaly score, product category, threshold, predicted defect type and the uploaded image.

Chapter 4

System Design

4.1 System Block Diagram

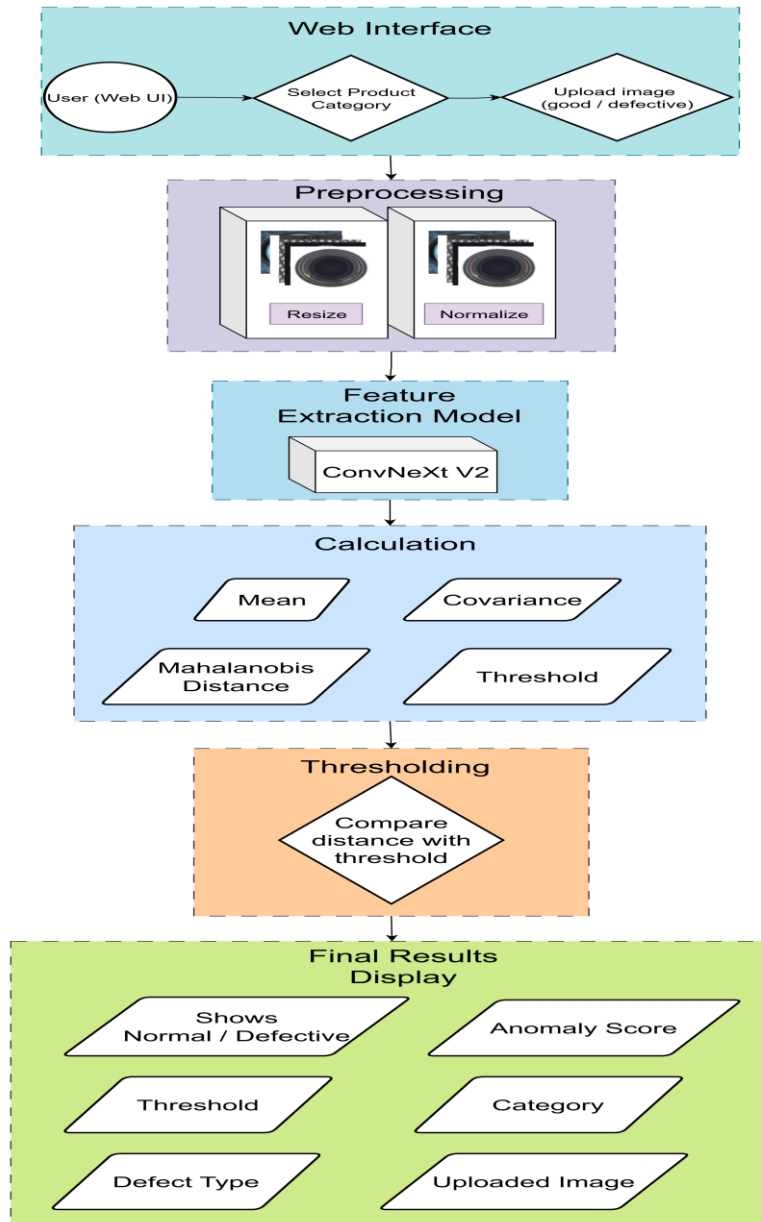


Figure 4.1 System Block Diagram

From the system block diagram above, there are several block modules including Web Interface Module, Preprocessing Module, Feature Extraction Module, Calculation Module, Threshold Module and Final Results Display Module. Each module has its own functions and responsibilities, and the process is done accordingly.

Web Interface Module

Real-time users go to inspection system web interface. Then the users are allowed to select the product category and choose the image that want to upload no matter it is good or defective image.

Preprocessing Module

Once the image had been uploaded, the image will undergo to preprocessing which transform the image by resizing and normalizing it.

Feature Extraction Module

Use the finetuned ConvNeXt V2 model to extract the feature of that certain image and compare it with the good features of image that extracted initially.

Calculation Module

Compute the mean and covariance of good features image to define the center of normal features and spread of correlations of normal features. Then will form a statistical model of normal feature. Next, compute the Mahalanobis Distance and threshold with optimized accuracy.

Threshold Module

In this module, compare the computed distance with threshold to determine whether the image is normal or defective image.

Final Results Display Module

After the analysis of image, the final result displays about the prediction of the image, anomaly score, threshold used, product category, predicted defect type and uploaded image.

4.2 System Components Specifications

4.2.1 Hardware

CPU	AMD Ryzen 7 6800H Processor
GPU	NVIDIA GeForce RTX 3060 6GB GDDR6
Memory	16GB DDR5-4800
Storage	512GB SSD M.2 2280

Table 4.2.1 Hardware Components

4.2.2 Software

Operating System	Windows 11
Frameworks	Pytorch, MMPretrain
Web Framework	Flask
Libraries	Torchvision, NumPy, PIL, scikit-learn, Utilities

Table 4.2.2 Software Components

4.2.3 Model Specifications

Model Used	ConvNeXt V2 Tiny
Input	224x224 RGB Image
Output	Feature vector (single 768-dimensional vector)

Table 4.2.3 Model Specifications



4.4 System Components Interaction Operation

1. User Interaction

- During the input layer, users are responsible to select a product category first before they are allowed to upload an image through the web interface.

2. Preprocessing

- The image that uploaded by the user will undergo to the preprocessing phase which includes resizing and normalizing the image.

3. Feature Extraction

- During feature extraction, the finetuned ConvNeXt V2 is responsible to extract the features vector of processed image that uploaded by the user. Then continue to compute the Mahalanobis Distance and threshold with optimized accuracy.

4. Anomaly Detection

- Once the calculation had been completed, the model will compare between the Mahalanobis distance and stored statistical model of good features which means that smaller distance will be similar to good features while larger distance will be far from good features.

5. Decision

- The pre-computed threshold will be used as cutoff to decide the image is normal or defective image. If higher than threshold will consider as defective image, if lower than threshold will consider as normal image.

6. Output

- Once the model predicted the image type, it will display the overall results which includes prediction of that image, anomaly score, threshold used, product category, predict defect type and uploaded image.

Chapter 5

System Implementation

5.1 Hardware Setup

Table 5.1 Hardware Setup

Description	Specifications
Model	Lenovo Legion 5 15ARH7H
Processor	AMD Ryzen 7 6800H Processor
Operating System	Windows 11
Graphic	NVIDIA GeForce RTX 3060 6GB GDDR6
Memory	16GB DDR5-4800
Storage	512GB SSD M.2 2280

5.2 Software Setup

Table 5.2 Software Setup

Description	Specifications
Platform	Visual Studio Code, GitHub
Dataset	MVTec Anomaly Detection Dataset
Frameworks	Pytorch, MMPretrain
Web Framework	Flask
Libraries	Torchvision, NumPy, PIL, scikit-learn, Utilities

5.3 Setting and Configuration

5.3.1 Model Fine-tuning

A pretrained ConvNeXt V2 Tiny was fine-tuned on the MVTec dataset and saved into another pth file called “finetuned.pth”

Configuration:

- Input image size: 224 x 224
- Optimizer: Adam
- Learning Rate: 0.0001
- Epochs: 20
- Batch size: 16

```

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = get_model(
    "convnext-v2-tiny_fcmae-in21k-pre_3rdparty_in1k",
    pretrained=True,
    head=dict(num_classes=num_classes)
).to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=1e-4)

```

Figure 5.3.1.1 Pretrained Model Retrieved

```

if val_acc > best_val_acc:
    best_val_acc = val_acc
    torch.save(model.state_dict(), "finetuned.pth")

```

Figure 5.3.1.2 Fine-tuned model checkpoint

5.3.2 Feature Extraction

After fine-tuning, only “good” training images were passed into the finetuned model by extracting the feature vector (768-D) that are normalized. Then the good features will be stored in “good_features_finetuned.pt” for future use.

```

out_file = "good_features_finetuned.pt"

```

Figure 5.3.2.1 Good Features Saved File

```

with torch.no_grad():
    for imgs, labels in tqdm(loader, desc="Extracting"):
        imgs = imgs.to(device)
        feats = model.extract_feat(imgs)
        if isinstance(feats, tuple):
            feats = feats[-1]
        if feats.dim() == 4:
            feats = F.adaptive_avg_pool2d(feats, 1).squeeze(-1).squeeze(-1)
        feats = F.normalize(feats, dim=1)
        all_feats.append(feats.cpu())
        all_labels.append(labels)

all_feats = torch.cat(all_feats) # [N, C]
all_labels = torch.cat(all_labels) # [N]

torch.save({
    "features": all_feats,
    "labels": all_labels,
    "class_to_idx": class_to_idx,
}, out_file)

print(f"✅ Saved {all_feats.shape[0]} feature vectors "
      f"({all_feats.shape[1]}p) to {out_file}")

```

Figure 5.3.2.2 Feature Vectors Extracted

5.3.3 Evaluation

Model performance is evaluated on MVTec testing dataset which contains both “good” and “defective” images. During the evaluation, thresholds for Mahalanobis distance are precomputed and saved in “finetuned.json” file. Once the threshold has been calculated, the model will start to analyze all testing datasets and show all metrics such as accuracy, recall, precision, F1-score and confusion matrix to evaluate the model’s performance.

```

threshold_file = 'finetuned.json'
if os.path.exists(threshold_file):
    with open(threshold_file, 'r') as f:
        threshold_dict = json.load(f)
    print("✅ Loaded existing thresholds from finetuned.json")
else:
    print("🔍 Computing best thresholds for each category...")
    threshold_dict = {}

```

Figure 5.3.3.1 Threshold Saved File



Figure 5.3.3.2 Example Evaluation Performance Metrics

5.3.4 Flask Web Interface

Flask web framework provides the functionality of image uploading and classification.

Configuration:

- Folder upload: uploads/
- Allowed extension images: .jpg, .jpeg, .png
- Hosted at: <http://0.0.0.0:5000>
- FrontEnd: index.html

```
UPLOAD_FOLDER = Path("uploads")
UPLOAD_FOLDER.mkdir(parents=True, exist_ok=True)
ALLOWED_EXT = {"png", "jpg", "jpeg"}
```

Figure 5.3.4.1 Folder Upload & Extension Images

```
if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=5000)
```

Figure 5.3.4.2 Local Hosting

```
return render_template(
    "index.html",
    categories=category_to_feats.keys(),
    result=result,
    anomaly=f"{dist:.4f}",
    threshold=f"{threshold:.4f}",
    category=category,
    defect_type=defect_type,
    image_url=url_for("uploaded_file", filename=filename)
)
```

Figure 5.3.4.3 Front End Index Template

5.4 System Operation (with Screenshot)

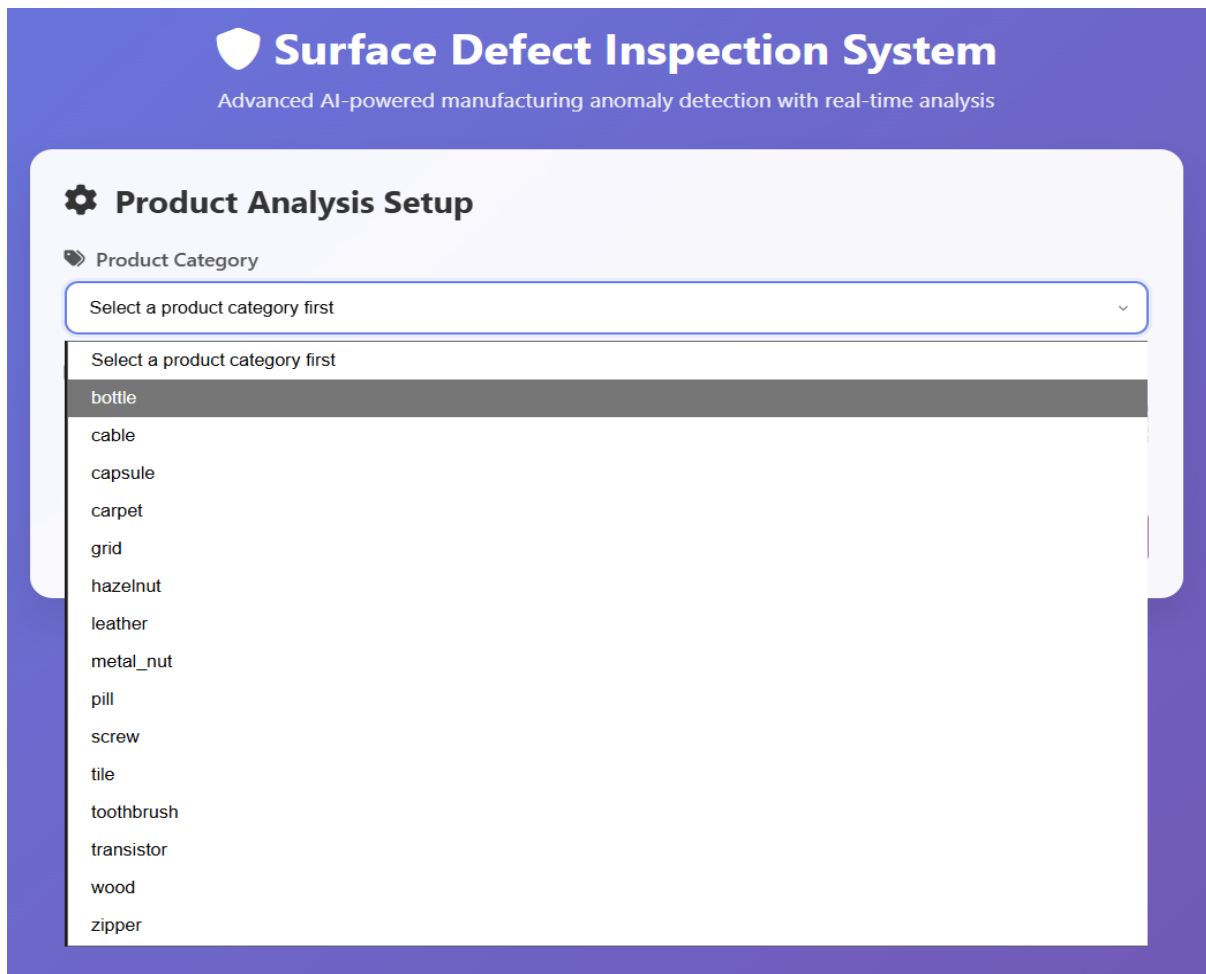


Figure 5.4.1 Product Category List

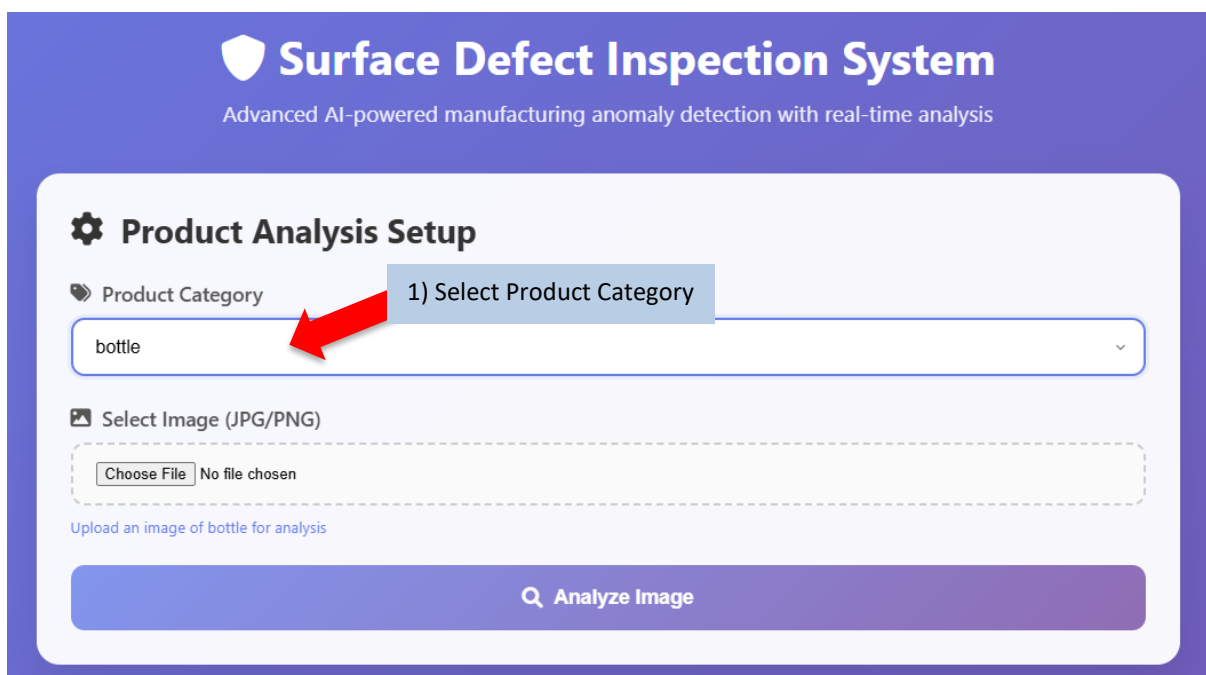


Figure 5.4.2 Select Product Category

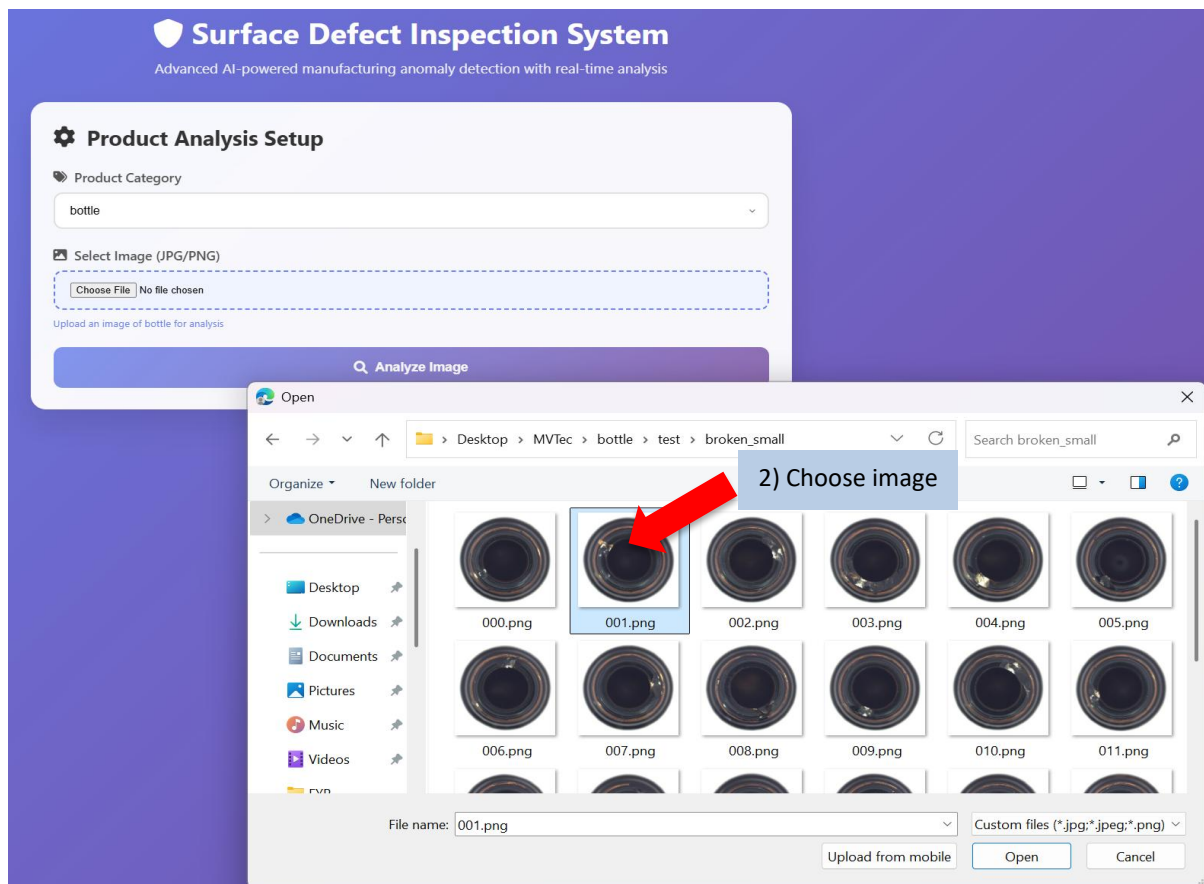


Figure 5.4.3 Upload Image from Same Category

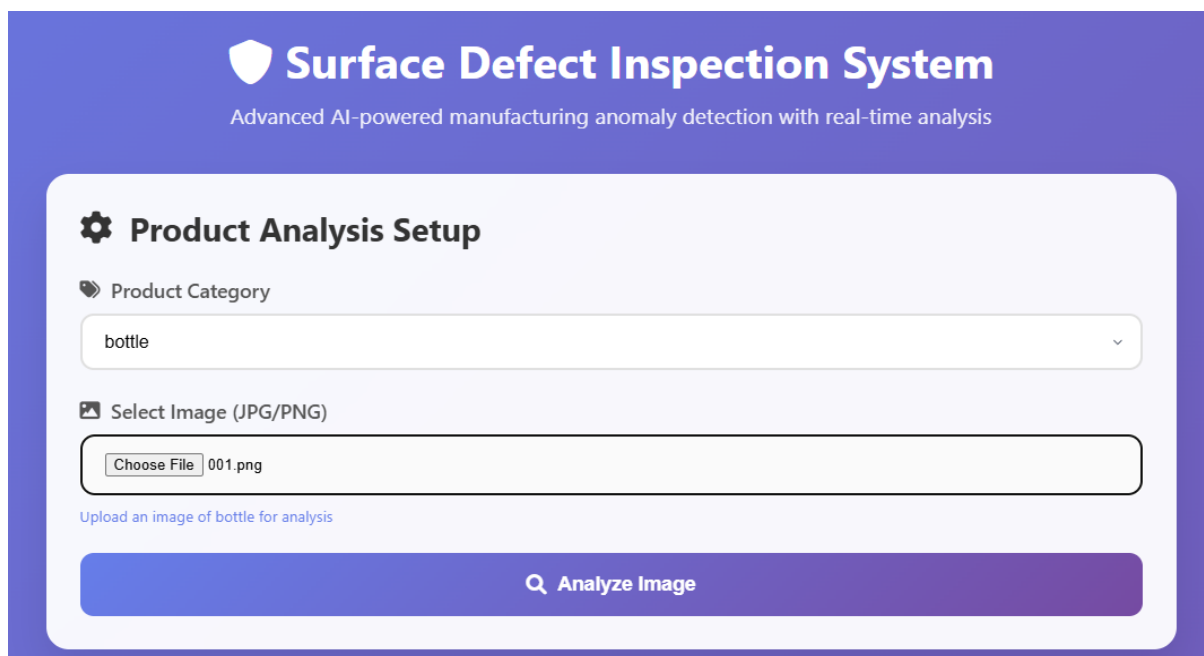


Figure 5.4.4 Product Category & Image Uploaded

Surface Defect Inspection System

Advanced AI-powered manufacturing anomaly detection with real-time analysis

Product Analysis Setup

Product Category

Select a product category first

Select Image (JPG/PNG)

Choose File

No file chosen

Please select a product category first to enable image upload

Analyze Image

3) Results

Analysis Result

Defective

Anomaly Score:

379.4351

Anomaly Level

100%

Threshold:

97.1264

Category:

bottle

Defect Type:

bottle_broken_small

Selected Image

Figure 5.4.5 Analysis Result

5.5 Implementation Issues and Challenges

With my GPU NVIDIA GeForce RTX 3060 6GB GDDR6, the **size of my pretrained model used needs to be very small** due to my **limited GPU memory**. Although ConvNeXt V2 has the model size until extra large (xLarge), but my GPU memory **only able support to tiny** model size. Therefore, this issue would **cause the fine-tuning process to become slower** and **lower accuracy** compared to other model size. Additionally, if many users send images at once, there is **high possibility which will make the GPU VRAM overflow and cause out of memory error**.

Following with this issue, the feature extraction would be **difficult and inaccurate** even though the model had been fine-tuned, then cause **low accuracy on detecting anomalies** while using the inspection system.

Next, my application expects the user to select a category that already exists in “good_features_finetuned.pt”. If the HTML form **sends an unknown category which does not exist in current trained categories**, the application will **show unexpected error, giving false result, rejecting the process or system crash**. In the meanwhile, the pre-computed threshold only consists of the trained categories, **if the threshold file (“finetuned.json”) does not have the threshold for unknown categories**, the model will **silently hide the error by always predicting the images as “normal”**. In general, when the product category selected and the image uploaded are **different in category**, the Mahalanobis distance will be in **wrong stats** and **used wrong threshold** to make decision, then **cause unreliable anomaly detection**.

5.6 Concluding Remark

In conclusion, the surface defect inspection system was successfully **implemented using a fine-tuned ConvNeXt V2 Tiny model** integrated into a Flask web interface to perform the anomaly detection. The hardware and software configuration setup **gave a robust environment to allow the system to perform model training, fine-tuning, feature extraction, evaluation, and real-time inference**. In the end, the system demonstrates its **ability to classify the image uploaded** as either normal or defective image by showing the image distance score and defect type.

However, there are **several limitations engaged during implementation**. Due to the **limited 6GB GPU memory of the NVIDIA RTX 3060**, only the **Tiny model size of ConvNeXt V2** could be used to do the fine-tuning process, which could **limit the accuracy** while learning the features compared to larger model sizes. GPU memory shortage also **brings the challenges when handling multiple user requests at once**, which could **cause the VRAM overflow and inference delays**. Additionally, the inspection system **relies strictly on pre-defined categories in both good features and threshold files**. Once the **mismatch between user input, feature vector, thresholds occurred**, could lead to **system error**, misclassification or unreliable anomaly detection outputs.

Despite these limitations, the implementation proved that it is **still possible to deploy an anomaly detection system** while using limited hardware resources like 6GB GPU memory through web-based interface by using Flask. Therefore, **address the GPU limited memory and enhance the category validation** is the **prior** things that need to be improved **to achieve more robust and scalable performance**.

Chapter 6

System Evaluation and Discussion

6.1 System Testing and Performance Metrics

My surface defect inspection system was tested on the MVTec AD testing dataset, which includes both normal and defective images. In order to measure the model's performance, standard machine learning evaluation metrics were applied such as:

- **Accuracy** – measures the proportion of correct classifications.
- **Precision** – measures how many predicted defective images were actually defective.
- **Recall (Sensitivity)** – measures how many actual defective images were correctly detected.
- **F1-Score** – harmonic mean of precision and recall for balanced evaluation.
- **Confusion Matrix** – provides detailed insights into true positives, false positives, true negatives, and false negatives.

In my code defines the “good” images as “0” while “defective” images as “1” because this inspection system is developed mainly for detecting on anomalies through the image. By using the scikit-learn's function which includes `accuracy_score`, `precision_score`, `recall_score` and `f1_score`, the score value will always label “1” as the positive class by default. Therefore, the “defective” images will define as positive class.

```
9 from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
```

Figure 6.1.1 Performance Metrics

```
class TestDataset(Dataset):
    def __init__(self, root_dir, transform, category):
        self.samples = []
        self.labels = []
        self.transform = transform
        self.category_path = os.path.join(root_dir, category, 'test')

        for defect_type in os.listdir(self.category_path):
            label = 0 if defect_type == 'good' else 1
            defect_path = os.path.join(self.category_path, defect_type)
            for fname in os.listdir(defect_path):
                if fname.lower().endswith(('.png', '.jpg', '.jpeg')):
                    self.samples.append(os.path.join(defect_path, fname))
                    self.labels.append(label)
```

Figure 6.1.2 Data Labeling

6.2 Testing Setup and Result

Setup:

- Hardware: Lenovo Legion 5 (AMD Ryzen 7 6800H, NVIDIA RTX 3060 6GB, 16GB RAM, 512GB SSD).
- Dataset: MVTec Anomaly Detection dataset.
- Frameworks: PyTorch, MMAPretrain.
- Web Interface: Flask.

Results:

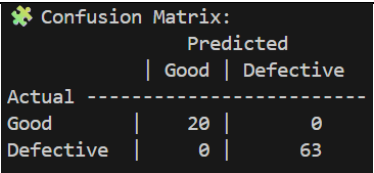
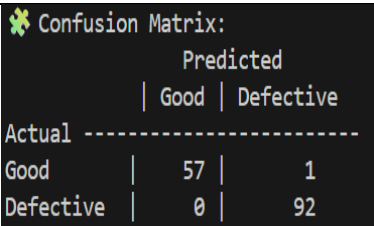
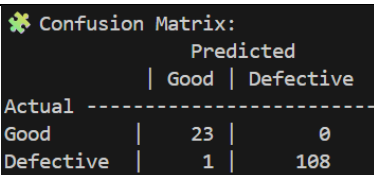
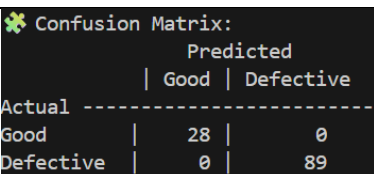
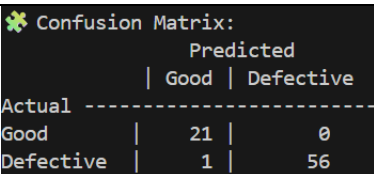
Product Category	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Confusion Matrix
Bottle	100.00	100.00	100.00	100.00	 <p>Figure 6.2.1 Bottle Matrix</p>
Cable	99.33	98.92	100.00	99.46	 <p>Figure 6.2.2 Cable Matrix</p>
Capsule	99.24	100.00	99.08	99.54	 <p>Figure 6.2.3 Capsule Matrix</p>
Carpet	100.00	100.00	100.00	100.00	 <p>Figure 6.2.4 Carpet Matrix</p>
Grid	98.72	100.00	98.25	99.12	

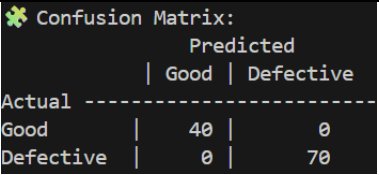
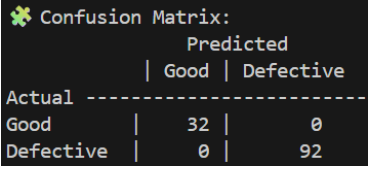
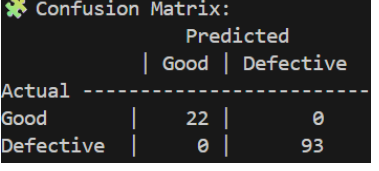
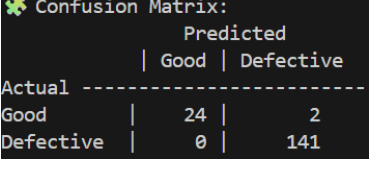
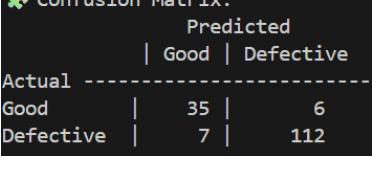
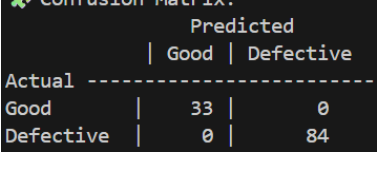
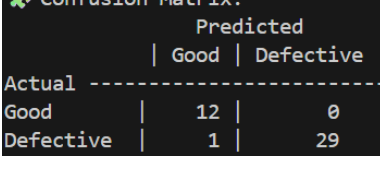
					Figure 6.2.5 Grid Matrix
Hazelnut	100.00	100.00	100.00	100.00	 <p>Figure 6.2.6 Hazelnut Matrix</p>
Leather	100.00	100.00	100.00	100.00	 <p>Figure 6.2.7 Leather Matrix</p>
Metal Nut	100.00	100.00	100.00	100.00	 <p>Figure 6.2.8 Metal Nut Matrix</p>
Pill	98.80	98.60	100.00	99.30	 <p>Figure 6.2.9 Pill Matrix</p>
Screw	91.88	94.92	94.12	94.51	 <p>Figure 6.2.10 Screw Matrix</p>
Tile	100.00	100.00	100.00	100.00	 <p>Figure 6.2.11 Tile Matrix</p>
Toothbrush	97.62	100.00	96.67	98.31	

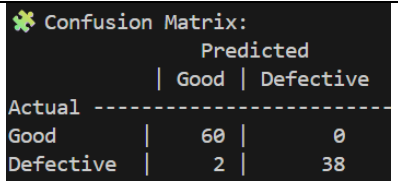
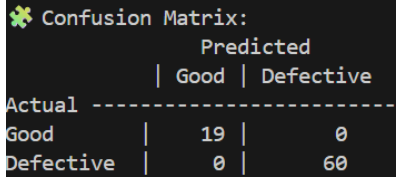
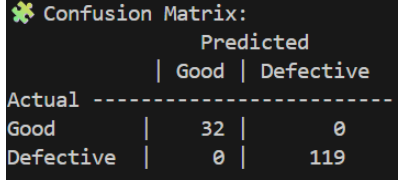

					Figure 6.2.12 Toothbrush Matrix
Transistor	98.00	100.00	95.00	97.44	 <p>Figure 6.2.13 Transistor Matrix</p>
Wood	100.00	100.00	100.00	100.00	 <p>Figure 6.2.14 Wood Matrix</p>
Zipper	100.00	100.00	100.00	100.00	 <p>Figure 6.2.15 Zipper Matrix</p>


Table 6.2 Evaluation Results

Web Interface Result:




Surface Defect Inspection System

Advanced AI-powered manufacturing anomaly detection with real-time analysis




Product Analysis Setup



Product Category

Select a product category first

▼




Select Image (JPG/PNG)

Choose File


No file chosen

Please select a product category first to enable image upload

Q Analyze Image



Analysis Result



Defective

⚡ Anomaly Score:

94.1434

Anomaly Level

100%

🎯 Threshold:


47.9826

🏷️ Category:

transistor

🔍 Defect Type:

transistor_damaged_case



Selected Image

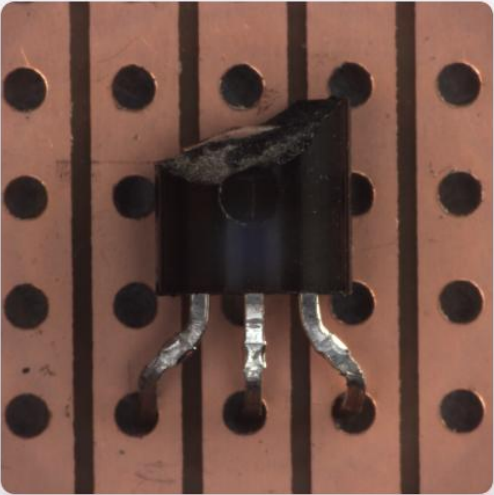


Figure 6.2.16 Web Interface Result

Bachelor of Information Technology (Honours) Communications and Networking
Faculty of Information and Communication Technology (Kampar Campus), UTAR

45

6.3 Project Challenges

During the project development process, there are some challenges occurred. The first challenges are time consuming on testing different model sizes because I required to test out the fine-tuning time consumption and the accuracy for each model size to ensure that the project can proceed to next step without any errors.

Besides that, the limitation of GPU memory also one of the challenges occurred during my project. Since the inspection system is using ConvNeXt V2 pretrained model from GitHub to do the fine-tuning on MVTec AD dataset, so download the pretrained model first is a must before do the fine-tuning. However, ConvNeXt V2 pretrained model contains several model sizes such as tiny, small, base, large and xlarge. Due to limited GPU memory, the inspection system can only choose the tiny size of ConvNeXt V2 pretrained model to do the fine-tuning so that the GPU memory would not get overflow and cause any unexpected error. If the larger model size of ConvNeXt V2 being chosen, the fine-tuning process might be slower or get cut off because lack of GPU memory. In addition, since the tiny model size is being used, the fine-tuning process is affordable but at some point, it reduced the precision of learned features because less capacity to learn, noisy gradients and the image resolution lower cause less detail.

There is another challenge which is about the category mismatch. While the inspection system doing the process, it will extract the good features from good training datasets first then proceed to do the evaluation on testing dataset. Furthermore, when doing the evaluation, the system will calculate the threshold for each category which could bring the optimized performance with better accuracy while doing decision on anomaly detection. Therefore, if the users input category that is not the same with the image from web interface, it could cause errors, unreliable detection because the system may use the wrong threshold or recognize the wrong category. Additionally, if there is unknown category image being input through the web interface, it also can cause the system failure as there are no pre-computed good features and threshold for that image category and may cause the system to always predict the image as "normal" in results.

6.4 Objectives Evaluation

- **To develop a deep learning model for surface defect inspection using an anomaly detection approach.**

A surface defect inspection system was successfully developed using ConvNeXt V2 model and adapt on the MVTec dataset. The anomaly detection approach is used to classify the images as "normal" or "defective" by implementing the feature-based extraction with Mahalanobis Distance to get robust anomaly detection instead of using simple classification on image.

- **To fine-tune a ConvNeXt-based CNN model.**

A pretrained ConvNeXt V2 tiny model is used to fine-tune on multiple categories of MVTec dataset. Due to the limited GPU memory, the tiny model size is the only affordable choice for fine-tuning. Avoid using the other model size such as small, base, large, xlarge even though they could potentially improve accuracy. Despite this, the tiny model size is enough to achieve high accuracy and generalization performance across multiple categories while GPU memory is limited. Then the fine-tuned model was saved as "finetuned.pth".

- **To extract and utilize good sample features for anomaly detection.**

The fine-tuned model is used to extract the good features from good samples and proceed to image preprocessing which include resizing and normalizing. The samples are resized into 224x224 resolution and being normalized into 768-dimensional feature vector. After that, these feature vectors were stored in "good_features_finnetuned.pt" then started to compute mean and covariance for each category. Next, use Mahalanobis Distance as an inference to differentiate the defective samples.

- **To evaluate the model's performance.**

After completed the fine-tuning and feature extraction process, the evaluation on model's performance was conducted on MVTec testing dataset that included both "good" and "defective" samples for all categories. Before proceed to performance metrics, threshold for all categories were calculated by optimizing the accuracy for each category since every category has different characteristics and stored in "finetuned.json". Then performance metrics such as accuracy, precision, recall, F1-score and confusion matrix were reported through terminal. Therefore, above results show the robustness of fine-tuned ConvNeXt V2 model on MVTec dataset with well performance while detecting anomalies from images.

- **To deploy the system with a web-based interface.**

The inspection system successfully deployed on web-based interface by using Flask template as platform to allow the users to upload images, interact by selecting the product category which consists of drop-down selection for users to choose and analyze the image anomaly detection results. The result include prediction result on uploaded image, anomaly score, threshold used, product category, predicted defect type and uploaded image. The overall web interface is well-designed to give excellent interaction and experience towards the users with web interface.

6.5 Concluding Remark

In conclusion, the surface defect inspection system with anomaly detection approach was successfully integrated with fine-tuned ConvNeXt V2 tiny model size by using Flask web interface to allow the users to classify "good" and "defective" images from the MVTec dataset. The inspection system achieved high performance metrics, demonstrating its capability as a lightweight inspection tool. However, challenges such as GPU memory limitations and category mismatches highlight areas for further enhancement. Therefore, future work should focus on optimizing the model for efficiency and flexibility, improving the input validation, and conducting better resource management to avoid any unreliable detection occur in real-world industrial applications.

Chapter 7

Conclusion and Recommendations

7.1 Conclusion

In conclusion, this project successfully developed a **Surface Defect Inspection System with Anomaly Detection Approach** using a fine-tuned ConvNeXt V2 Tiny model and deployed it through a lightweight Flask-based web interface. The system was able to do the classification and detection between "good" and "defective" samples with high accuracy across multiple product categories. The integration of **feature extraction from good samples** and the use of **Mahalanobis Distance with optimized thresholds** provided reliable anomaly detection performance, while the web interface ensured practical usability for real-time users or industrial applications.

The results indicates that the developed inspection system is effective on solving the limitations of traditional manual inspection, offering improved accuracy, reliability, and efficiency. Even though the inspection system implemented the fine-tuned Tiny model of ConvNeXt V2 which could reduce the accuracy during learning process due to **GPU memory limitations**, the system still achieved strong evaluation metrics on accuracy, precision, recall, and F1-score on MVTec testing dataset. Additionally, the confusion matrix also included to analyze whether is there any misclassification occurred during evaluation.

However, there are several challenges were identified such as limited GPU memory could restricted the model size to be implemented and its scalability. Next, while unknown category or mismatch happened, a potential system errors and unreliable predictions could occur because the threshold used is different due to mismatch category or the threshold will be empty and always predict the image as "normal". Besides that, if the inspection system handled multiple requests at once it can have high possibility to cause memory overflow and slow inference time.

In conclusion, the system has proven its feasibility and effectiveness as a lightweight and scalable anomaly detection tool in manufacturing quality control, while also highlighting key areas for future enhancement.

7.2 Recommendations

From the overall project progress and the review on inspection system, there were some limitations from the system. In order to further improve the system and extend its applicability into industrial environments, there are some recommendations that can be used such as:

1. Hardware and Model Optimization

- Utilize GPUs with larger memory capacity to allow the use of larger ConvNeXt variants without exceeding hardware limits.
- Implement **batch inference optimization** to handle multiple requests efficiently.

2. Category Validation and Error Handling

- Implement stronger **input validation mechanisms** to ensure that uploaded images match the selected product category, reducing errors caused by category mismatch.
- Apply error handling to notify users that the category mismatch to reselect category and reupload the images.
- Apply error handling to notify users when undefined categories are uploaded.

3. Scalability and Deployment

- Deploy the system on **cloud platforms with multiple GPU support**, enabling industrial-scale real-time inspection without causing any hardware limitation problems.

4. Expanded Dataset and Model Generalization

- Expand training and fine-tuning on **additional industrial datasets** to improve robustness against unseen defect types.
- Considering multiples approaches by applying different methods such as supervised, unsupervised, self-supervised, reinforcement and others learning techniques based on different industrial environments [12].

5. User Experience Design

- Enhance the web interface with more interactive features such as **real-time visualization of defect parts, performance analytics dashboards, real-time fine-tuning button** and others.


REFERENCES

- [1] T. I. Systems, “The Difference Between Traditional Inspection Methods and Vision Inspection Systems,” *Trident Information Systems Pvt Ltd*, Aug. 02, 2023. <https://tridentinfo.com/the-difference-between-traditional-inspection-methods-and-vision-inspection-systems/>
- [2] I. Farady, B. Khimsuriya, R. Sagathiya, P. -C. Lin and C. -Y. Lin, "Implementing Multi-Level Features in a Student-Teacher Network for Anomaly Detection," 2023 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), PingTung, Taiwan, 2023, pp. 839-840, doi: 10.1109/ICCE-Taiwan58799.2023.10226695.
- [3] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, “MVTec AD -- A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection,” *Thecvf.com*, pp. 9592–9600, 2019
- [4] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked Autoencoders Are Scalable Vision Learners,” *arXiv:2111.06377 [cs]*, Dec. 2021, Available: <https://arxiv.org/abs/2111.06377>
- [5] Withinpixels, “Home - saiwa,” *saiwa*, Feb. 15, 2024. <https://saiwa.ai/app/detection/anomaly-detection/> (accessed Aug. 28, 2024).
- [6] Y. Chen, Y. Ding, F. Zhao, E. Zhang, Z. Wu, and L. Shao, “Surface Defect Detection Methods for Industrial Products: A Review,” *Applied Sciences*, vol. 11, no. 16, p. 7657, Jan. 2021, doi: <https://doi.org/10.3390/app11167657>.
- [7] T. Peng, Y. Zheng, L. Zhao, and E. Zheng, “Industrial Product Surface Anomaly Detection with Realistic Synthetic Anomalies Based on Defect Map Prediction,” *Sensors*, vol. 24, no. 1, pp. 264–264, Jan. 2024, doi: <https://doi.org/10.3390/s24010264>.
- [8] P. M. Bhatt *et al.*, “Image-Based Surface Defect Detection Using Deep Learning: A Review,” *Journal of Computing and Information Science in Engineering*, vol. 21, no. 4, Feb. 2021, doi: <https://doi.org/10.1115/1.4049535>.

REFERENCES

- [9] I. Farady, C.-C. Kuo, H.-F. Ng, and C.-Y. Lin, "Hierarchical Image Transformation and Multi-Level Features for Anomaly Defect Detection," *Sensors*, vol. 23, no. 2, p. 988, Jan. 2023, doi: <https://doi.org/10.3390/s23020988>.
- [10] "ConvNext - A ConvNet for the 2020s (2022) - Hugging Face Community Computer Vision Course," *Huggingface.co*, 2020. <https://huggingface.co/learn/computer-vision-course/en/unit2/cnns/convnext> (accessed Sep. 08, 2025).
- [11] K. Lee, K. Lee, H. Lee, and J. Shin, "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks," *arXiv.org*, Oct. 27, 2018. <https://arxiv.org/abs/1807.03888>
- [12] L. Ruff *et al.*, "A Unifying Review of Deep and Shallow Anomaly Detection," *arxiv.org*, Sep. 2020, doi: <https://doi.org/10.1109/JPROC.2021.3052449>.

POSTER



UTAR
UNIVERSITI TUNKU ABDUL RAHMAN


**FACULTY OF INFORMATION
COMMUNICATION AND TECHNOLOGY**

Bachelor of Communications and Networking (HONOURS)

Surface Defect Inspection System

WITH ANOMALY DETECTION APPROACH

Developed by: Androw Ling Kian Quan
Supervisor: Dr Fityanul Akhyar
Moderator: Ts Dr Gan Ming Lee



INTRODUCTION

An inspection system is important to detect the anomalies in industrial application. Therefore, using deep learning and computer vision techniques able to enhance the accuracy on detecting anomalies and efficiency at the same time.

PROJECT

OBJECTIVE

- Develop deep learning model
- Fine-tune ConvNeXt-based CNN model
- Good Features Extraction
- Evaluate model performance
- Deploy to Front End web interface for practical usability

METHODOLOGY

- 1 Fine-Tuning
- 2 Good Features Extracted
- 3 Performance Evaluation
- 4 Deployment on Web

WEB INTERFACE RESULT

