國立陽明交通大學
智慧與綠能產學研究所
碩士論文

Institute of Smart Industry and Green Energy
National Yang Ming Chiao Tung University
Master Thesis

具備不確定性感知的高斯過程與交叉熵混合模型在瑕疵分類中的應用

Defect Classification with Indeterminate Awareness Using Gaussian Process and Cross-Entropy Hybrid Models

研究生 : 林怡瑄 (Lin, Yi-Xuan)
指導教授 : 馬清文 (Ma, Ching-Wen)

中華民國 一一四年一月
January 2025

具備不確定性感知的高斯過程與交叉熵混合模型在瑕疵分類中的應用

Defect Classification with Indeterminate Awareness Using Gaussian Process and Cross-Entropy Hybrid Models

研 究 生：林怡瑄　　Student: Lin, Yi-Xuan

指導教授：馬清文 博士　　Advisor: Dr. Ma, Ching-Wen

國立陽明交通大學

智慧與綠能產學研究所

碩士論文

A Thesis
Submitted to Institute of Smart Industry and Green Energy
College of Artificial Intelligence and Green Energy
National Yang Ming Chiao Tung University
in partial Fulfilment of the Requirements
for the Degree of
Master of Science
in
Artificial Intelligence

January 2025

Taiwan, Republic of China

中華民國 一一四年一月

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Ma, for his invaluable guidance, encouragement, and unwavering support throughout the course of my research and study. His profound knowledge and insightful suggestions have been instrumental in shaping this thesis.

I am also immensely thankful to my fellow lab members for their camaraderie, collaboration, and countless discussions that have enriched my academic journey. Their support and feedback have been vital in overcoming challenges and achieving progress.

Lastly, I am deeply grateful to my family and friends for their constant encouragement, understanding, and love. Their belief in me has been a source of strength and inspiration throughout this journey.

具備不確定性感知的高斯過程與交叉熵混合模型在瑕疵分類中的應用

學生：林怡瑄　　指導教授：馬清文 博士

國立陽明交通大學 智慧與綠能產學研究所

摘　　要

隨著工業製程的不斷複雜化，對元件缺陷進行準確、高效的檢測與分類已成爲一項重要挑戰。本研究旨在改進實驗室先前提出的瑕疵檢測系統架構，針對該系統在良品與瑕疵品的二元分類中已取得良好性能，但在精確分類各種缺陷類型時存在不確定性的問題，提出了創新解決方案。我們首先將原有的二元分類器（Good/Bad Classifier）替換爲具備四類別分類能力的模型，並將其嵌入混合專家架構的上分支，以實現對瑕疵種類的更精細分類。其次，我們結合高斯過程（Gaussian Process）與交叉熵（Cross Entropy）優化方法，設計了一種包含三個針對單一瑕疵類型的 GPCE 混合模型的引導式瑕疵檢測系統。針對上述方法，我們逐步分析其在瑕疵檢測系統中的優勢與局限性，爲開發更高效且穩健的瑕疵分類方法提供理論支持與實驗依據。

通過這種方法，本研究不僅提高了對不同瑕疵類型的分類能力，還通過不確定性量化技術，增強了系統對於未知樣本或異常情況的應對能力。初步結果顯示，所提出的改進方法能夠在保持整體檢測性能的同時，有效解決傳統分類器在特定瑕疵類別上的表現局限性，爲瑕疵檢測系統的智能化和高可靠性應用提供了新的可能性。

關鍵詞: 瑕疵分類、混合專家、不定性分析、深度學習、高斯過程。

Defect Classification with Indeterminate Awareness Using Gaussian Process and Cross-Entropy Hybrid Models

Student: Lin, Yi-Xuan Advisor: Dr. Ma, Ching-Wen

Institute of Smart Industry and Green Energy
National Yang Ming Chiao Tung University

Abstract

With the growing complexity of industrial manufacturing processes, accurate and efficient defect detection and classification remain significant challenges. This study enhances a previously proposed defect detection system by addressing its limitations in classifying specific defect types despite strong performance in binary classification. To this end, we replaced the binary classifier with a four-class model embedded in a hybrid expert framework for finer defect classification. Additionally, we developed a guided detection system combining Gaussian Process (GP) and Cross Entropy (CE) optimization, featuring three GPCE hybrid models tailored to specific defect types.

We systematically analyzed the advantages and limitations of these approaches, providing theoretical and experimental support for developing more efficient and robust defect classification methods. Preliminary results demonstrate improved classification performance and enhanced resilience to unknown samples through indeterminate quantification, offering promising applications for intelligent and reliable defect detection systems.

Keywords: Defect Classification, Hybrid Expert, Indeterminate Analysis, Deep Learning, Gaussian Process

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Defect classification is crucial for ensuring product quality and optimizing manufacturing processes. Although significant progress has been made in defect classification when datasets are comprehensive and balanced, real-world applications often face challenges due to limited data availability. These challenges include insufficient sample sizes or imbalanced class distributions, which hinder the development of reliable and accurate classification systems.

Our dataset reflects a notable imbalance between old and new component types. Old components refer to those that have been extensively represented in the training dataset, with ample samples of both normal and defect types. These components have well-established detection reliability due to their inclusion in prolonged training processes. However, new components were introduced more recently, resulting in fewer defect samples available during training. While this limits the comprehensiveness of training for defect detection, including new components expands the dataset and enhances the model's adaptability to the latest manufacturing processes and technologies. Table 1.1 (below) provides a summary of the sample distribution for old and new components during training and testing phases, highlighting the disparity in the availability of defect types and its implications for classification performance.

Table 1.1: Comparison of old and new components.

	Sample availability during training	Sample availability during testing
Old component	Normal type : many Defect type : many	Normal type : many Defect type : many
New component	Normal type : many Defect type : few or none	Normal type : many Defect type : enough

Our laboratory has previously addressed these dataset challenges in the context of defect detection, achieving robust results in distinguishing defective from non-defective components under such constraints. Building on this success, the current study extends these methodologies to defect classification, with the aim of providing a fine-grained categorization of defect types and addressing the unique challenges posed by the imbalance of the dataset.

1.2 Motivation

Defect detection technologies have shown significant potential to address practical challenges in industrial applications, such as identifying defective components and ensuring product quality. However, while these systems are effective in distinguishing between defective and non-defective samples, they lack the capability to provide detailed information about the specific types of defects. By extending defect detection to defect classification, it becomes possible not only to detect defective components but also to identify the nature of defects, enabling more targeted quality control measures, enhanced manufacturing diagnostics, and improved process optimization. This shift from detection to classification represents a crucial step toward intelligent and comprehensive defect management systems.

To achieve this goal, it is essential to address the challenges posed by datasets with inherent imbalances, particularly those involving differences between old and new components. These datasets often feature an abundance of samples for old components, while new components have significantly fewer defect samples, leading to challenges in building reliable and unbiased

classification models. This study focuses on developing a defect classification system capable of handling these disparities, ensuring accurate classification performance for both old and new components while maintaining robustness in various defect scenarios.

1.3 Goal

The primary objective of this study is to enhance the previously developed defect detection architecture in our laboratory by transitioning it into a more advanced defect classification system. While the current defect detection framework effectively identifies whether a sample is defective or non-defective, it lacks the capability to distinguish between specific types of defects. This limitation reduces its applicability in industrial scenarios where understanding the nature of the defect is critical for targeted quality control and process optimization.

Defect classification not only allows for the detection of defective components but also provides detailed insights into the underlying defect types. This finer-grained understanding enables manufacturers to identify root causes, implement corrective actions, and improve production processes more effectively. By transforming the existing defect detection framework into a classification system, we aim to address these challenges and expand the utility of the framework in real-world applications.

The improved system focuses on achieving high classification accuracy while maintaining robustness across diverse defect types, even under constraints such as the presence of new component types. This study explores innovative approaches, such as multi-class classification and hybrid modeling, to bridge the gap between defect detection and classification, making significant contributions to the field of intelligent quality control.

1.4 Contribution and Limitation

Contribution

This study makes the following key contributions to the field of defect classification and intelligent quality control:

1. **Implementation of Gaussian Process and Cross Entropy Hybrid Model**

   We successfully implemented a GPCE hybrid model that combines Gaussian Process (GP) and Cross Entropy (CE) optimization for defect classification tasks. This approach leverages the strengths of both methodologies, enabling the system to achieve enhanced classification performance and robustness across diverse defect types.

2. **Handling of Indeterminate Predictions**

   The proposed system effectively addresses scenarios where model uncertainty is excessively high or classification outcomes are inconsistent. Such cases are reliably categorized as indeterminate, preventing these samples from being misclassified into incorrect defect categories. This capability significantly reduces the risk of false positive or false negative errors, enhancing the reliability of the classification system in practical applications.

3. **Introduction and Analysis of Multi-classification and One-vs-Rest Methods**

   We proposed and systematically analyzed a multi-class classification strategy alongside a one-vs-rest approach. These methods provide a foundational framework for future research in defect classification, offering insights into their strengths, limitations, and practical implications for improving system performance in industrial contexts.

Chapter 2

Related Works

2.1 Defect Classification

Defect classification is a critical task in industrial quality control, ensuring the identification of defective components and the precise categorization of defect types. Traditional binary classifiers have been widely used to distinguish defective from non-defective items but often fail to address the complexity of multi-class scenarios. Such limitations reduce their effectiveness in diagnosing root causes and implementing corrective measures. Additionally, datasets frequently exhibit significant class imbalances, where underrepresented defect categories may lead to biased predictions favoring majority classes.

Several approaches have been proposed to address these challenges. Class rebalancing techniques, such as SMOTE [1] and Adaptive Synthetic Sampling [2], mitigate class imbalance by generating synthetic samples for minority classes. Cost-sensitive learning, which assigns misclassification penalties based on class importance, has been shown to improve performance in minority class predictions [3]. Deep learning methods, particularly CNNs, have demonstrated significant success in defect classification. For instance, Zhang et al.[4] achieved high accuracy in steel surface defect detection by leveraging hierarchical feature extraction in CNNs. Similarly, Huang et al.[5] proposed a multi-scale attention mechanism combined with CNNs to enhance defect classification robustness. Transfer learning has also emerged as an effective strategy, with pre-trained models fine-tuned on domain-specific datasets yielding improved results [6]. Active learning techniques [7] have further improved classification performance in scenarios with limited labeled data by focusing on labeling the most informative samples.

Despite these advancements, existing methods often struggle with datasets exhibiting significant differences between new and old component types. This study addresses these gaps by

employing a hybrid Gaussian Process and Cross Entropy model, integrating uncertainty quantification to enhance multi-class classification reliability and manage dataset imbalances.

2.2 Hybrid Models

Hybrid models have gained significant attention for their ability to integrate the strengths of different machine learning paradigms. These models are particularly useful in multi-task scenarios, such as defect type and component type classification, where the extracted features need to capture task-specific information.

Numerous studies have demonstrated the efficacy of hybrid models. Zhang et al.[8] combined CNNs for feature extraction with SVMs for classification, improving performance in imbalanced datasets. Wu et al.[9] proposed integrating RNNs with statistical models to handle time-series defect data, achieving enhanced generalization capabilities. Multi-task frameworks, such as the one developed by Sun et al.[10], enable the simultaneous classification of defect and component types, demonstrating improved generalization across diverse datasets. Hybrid ensemble methods, such as the combination of decision trees and deep neural networks proposed by Chen et al.[11], have effectively addressed complex defect classification tasks. Additionally, autoencoders combined with random forests [12] have shown effectiveness in managing imbalanced datasets while maintaining robust classification performance.

Building upon these approaches, our research incorporates Gaussian Processes into a hybrid modeling framework. This allows for the simultaneous classification of defect and component types while addressing dataset imbalances and managing uncertainty through probabilistic modeling.

2.3 Indeterminate Predictions

Indeterminate predictions, which occur when models exhibit high uncertainty, present a critical challenge in defect classification. Misclassifications under such circumstances can result in significant costs, making it essential to systematically handle ambiguous predictions. Effec-

tive uncertainty quantification ensures that these cases are reliably categorized, reducing risks in practical applications.

Several methods for uncertainty estimation have been proposed. Bayesian Neural Networks (BNNs), introduced by Gal and Ghahramani [13], estimate uncertainty using dropout-based Bayesian approximations. Ensemble methods, such as deep ensembles [14], have demonstrated robust uncertainty quantification. Gaussian Processes (GPs), as described by Rasmussen and Williams [15], offer probabilistic predictions with uncertainty measures and have been successfully applied to industrial defect detection [16]. Techniques like Monte Carlo Dropout [17] and variational inference [18] further enhance uncertainty estimation in deep learning models.

This study employs Gaussian Processes to systematically manage predictions by categorizing them as ”good,” ”defective,” or ”indeterminate.” Component-specific uncertainty thresholds ensure ambiguous predictions are effectively handled, improving system reliability.

2.4 Hyperparameter Optimization

Hyperparameter optimization (HPO) is essential for fine-tuning machine learning models to achieve optimal performance. Traditional methods, such as manual tuning, are inefficient for modern, high-dimensional models, necessitating the use of automated approaches.

Commonly used techniques include grid search and random search, where Bergstra and Bengio [19] demonstrated that random search is more efficient than grid search in exploring high-dimensional parameter spaces. Bayesian optimization [20] has emerged as a powerful method, iteratively refining hyperparameter configurations using probabilistic models. Gradient-based optimization [21] enables efficient optimization of differentiable hyperparameters, while evolutionary algorithms, such as genetic algorithms [22], have also been effectively applied to explore large-scale parameter spaces. Automated frameworks like Neural Architecture Search [23] further streamline hyperparameter tuning, offering optimized configurations for neural networks.

In this study, Bayesian optimization is applied to tune uncertainty thresholds and other hyperparameters, balancing computational efficiency and model performance in defect classification

tasks.

Chapter 3

Method

3.1 Model Architecture

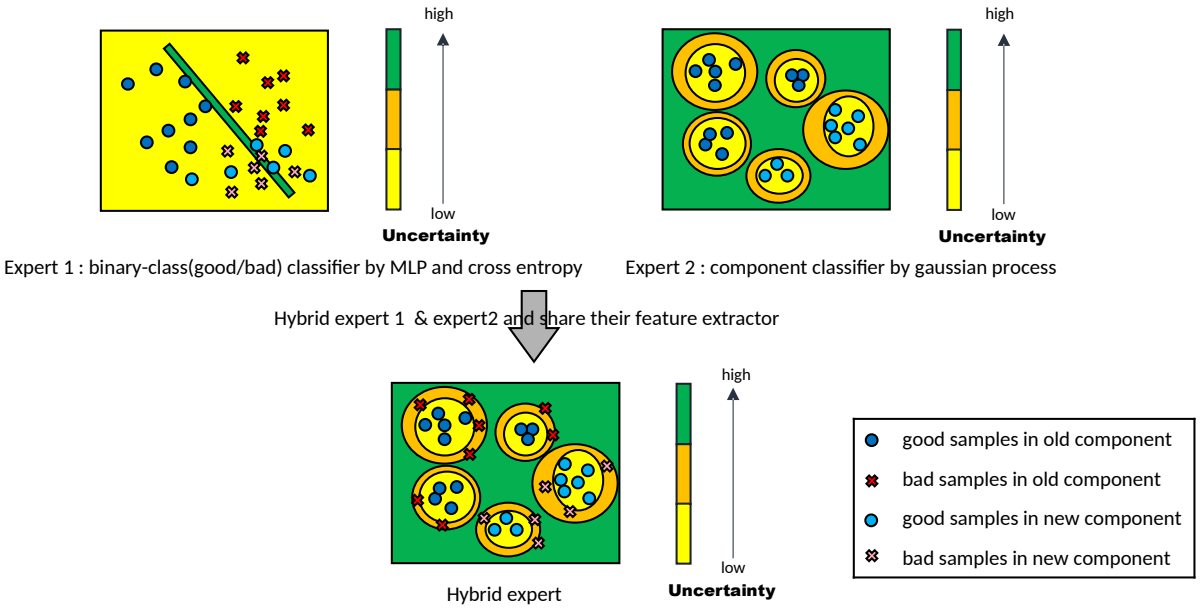


Figure 3.1: Illustration of GPCE hybrid model

This study proposes a hybrid expert architecture, named the Gaussian Process and Cross Entropy Hybrid Model (GPCE hybrid model), to address the challenges of achieving accurate and reliable defect classification in industrial applications. In traditional defect classification systems, classifiers often face confusion when dealing with insufficient defect samples in certain component categories. To mitigate this issue, we designed a hybrid architecture consisting of two expert models: Expert 1 focuses on binary classification of defective and non-defective components, while Expert 2 performs component type classification. The results from both experts are integrated to improve the overall performance of defect classification.

In this architecture, Expert 1 employs a Multi-Layer Perceptron (MLP) as its classifier and

is optimized using a cross-entropy loss function, providing outputs y_{def} for binary classification between defective and non-defective components. However, when defect data are insufficient or unevenly distributed, Expert 1 may overlook fine-grained features related to component types, leading to misclassification. To address this limitation, Expert 2 is introduced, leveraging Gaussian Processes (GPs) for component type classification and generating probabilistic outputs p_{com}. GPs are not only capable of capturing subtle features specific to each component type but also provide quantified uncertainty metrics that serve as additional decision-making support for the system. By leveraging the probabilistic distributions generated by GPs, the component classification results can be transformed into auxiliary estimates for defect classification, enhancing the system's ability to identify confused categories.

The overall architecture includes a shared feature extractor $f_\theta(x)$, which processes the input data x into feature representations z. These features are simultaneously passed to both Expert 1 and Expert 2 for defect and component type classification, respectively. Expert 1 outputs y_{def}, representing the binary classification results for defective and non-defective components, while Expert 2 outputs p_{com}, representing the probabilistic distributions of component type classifications. In addition, the GP classifier in Expert 2 generates uncertainty metrics U_p, which quantify the confidence of its predictions. The uncertainty distributions generated internally by the GP are processed through a transformer and then combined with the binary classification results from Expert 1 to make final classification decisions.

To further enhance classification reliability, we designed an uncertainty evaluation module that dynamically manages the system's confidence in its predictions. In this module, when the uncertainty of a sample is below a predefined threshold, Expert 2 classifies the sample as "non-defective." For samples with intermediate confidence, Expert 2 classifies them as "defective." When the uncertainty exceeds a specific threshold, the sample is classified as "undefined" to avoid misclassification. Finally, the results of Experts 1 and 2 are integrated to generate the final output, considering both defect classification and component classification, which collectively improve the system's overall accuracy and robustness.

This design fully leverages the probabilistic modeling capabilities of Gaussian Processes

and the classification efficiency of the Multi-Layer Perceptron. By incorporating uncertainty quantification, the model achieves fine-grained defect classification even in scenarios with insufficient defect data or imbalanced distributions across component categories. Moreover, the inclusion of uncertainty metrics enhances the system's robustness when handling unknown samples or anomalous conditions.

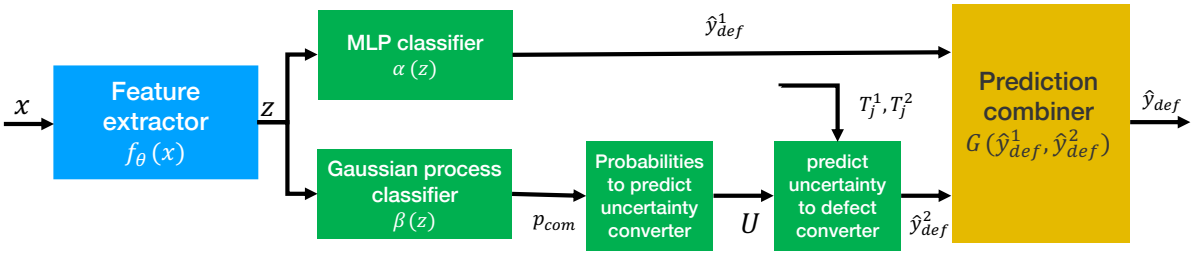


Figure 3.2: Illustration of GPCE hybrid model

Block Diagram Details:

The proposed system architecture, as illustrated in the block diagram, is designed to achieve accurate defect classification by integrating multiple components with complementary functionalities. Below, we provide a detailed description of each block in the architecture:

1. **Feature extractor,** $f_\theta(\cdot)$. The feature extractor is implemented using a pre-trained MobileNetV3 Large Backbone, which leverages ImageNet for pre-training. This component extracts high-level embedding features (z) from the raw input data (x). These features serve as the shared representation for both defect classification and component classification tasks, enabling a unified processing pipeline.

2. **MLP Classifier ,**$\alpha(z)$**:**. The Multi-Layer Perceptron (MLP) classifier is a fully connected layer designed specifically for binary classification tasks. It classifies the input features z into two categories: "good" and "defective" corresponding to non-defective and defective components. The output of this block is denoted as $\hat{y}^1_{def}$, which represents the binary decision of the MLP classifier.

3. **Gaussian Process Classifier,** $\beta(z)$. The Gaussian Process Classifier (GPC) is responsible for component type classification. By analyzing the embedding features (z), the GPC

outputs a probabilistic vector p_{com} over K component types. This probabilistic output captures subtle distinctions among component types, which can provide auxiliary information for defect classification.

4. **Probabilities-to-Predict Uncertainty Converter**. This module converts the probabilistic output (p_{com}) from the Gaussian Process Classifier into an uncertainty measure (U). The uncertainty quantifies the confidence of the classification results, providing a critical metric for identifying ambiguous or low-confidence predictions.

5. **Predict Uncertainty-to-Defect Converter, $U \rightarrow \hat{y}^2_{def}$.** The uncertainty measure(U)is further analyzed by this module to produce a defect classification result ($\hat{y}^2_{def}$), which classifies samples into one of three categories: "good", "defective", or "indeterminate". Predefined thresholds (T^1_j, T^2_j) guide this conversion, ensuring reliable handling of samples with varying levels of uncertainty:

$$\hat{y}^2_{def} = \begin{cases} \text{"good"}, & \text{if } U \leq T^1_j \text{ (low uncertainty)} \\ \text{"defective"}, & \text{if } T^1_j < U \leq T^2_j \text{ (moderate uncertainty)} \\ \text{"indeterminate"}, & \text{if } U > T^2_j \text{ (high uncertainty)} \end{cases} \tag{3.1}$$

6. **Prediction Combiner, $G(\hat{y}^1_{def}, \hat{y}^2_{def})$.** The Prediction Combiner integrates the outputs from the MLP classifier($\hat{y}^1_{def}$) and the defect classification result from the uncertainty-based module($\hat{y}^2_{def}$). By combining these results, the system achieves a more robust final prediction($\hat{y}_{def}$).This integration allows the system to leverage the complementary strengths of binary defect classification and uncertainty-aware decision-making, reducing the risk of misclassifications.

7. **Output, $\hat{y}_{def}$.** The final output of the system is a classification result that categorizes each sample as "good", "defective", or "indeterminate". This output reflects the combined predictions from the MLP and Gaussian Process classifiers, ensuring accurate and reliable defect classification.

Key Parameters:

z : Embedding features extracted by the feature extractor.

p_{com} : Probabilistic output from the Gaussian Process Classifier.

U : Uncertainty measure derived from (p_{com})through the Probabilities-to-Predict Uncertainty Converter.

T_j^1, T_j^2 : Thresholds for differentiating uncertainty levels.

3.2 a Hybrid Expert, incorporating both Multi-Defect-Type and Multi-Component-Type Classifier

In this study, we improved the upper branch of the GPCE hybrid model (Gaussian Process and Cross Entropy Hybrid Model) by redesigning it as a four-class classifier. This classifier is capable of distinguishing among four categories: ”good”, ”shift”, ”broke” and ”short”. Meanwhile, the lower branch retains its original functionality of estimating three categories: ”good”, ”defective” and ”indeterminate”. This dual-layer classification strategy allows the model to accurately classify defect types while effectively managing samples with high uncertainty, thereby enhancing the overall reliability of the classification system.

In the upper branch, the four-class classifier focuses on fine-grained defect classification and is optimized using a cross-entropy loss function to handle multi-class classification tasks. The lower branch, based on a Gaussian Process, not only generates probabilistic distributions for sample predictions but also quantifies the uncertainty of the classification results. This uncertainty quantification strategy enables the model to flag low-confidence samples, thereby avoiding misclassification. Additionally, the feature extractor, which is shared across both branches, efficiently extracts both global and local features from the input samples, supporting the classification tasks of both branches.

The core contribution of this architecture lies in its effective combination of defect classification and uncertainty handling. The four-class classifier in the upper branch improves the model’s ability to differentiate among multiple defect types, while the lower branch provides ad-

ditional decision support by marking high-risk samples as ”indeterminate”, thus mitigating the risk of misclassification. This dual-layer design not only enhances the model’s classification accuracy but also improves its robustness in handling imbalanced data distributions.

Training Procedure

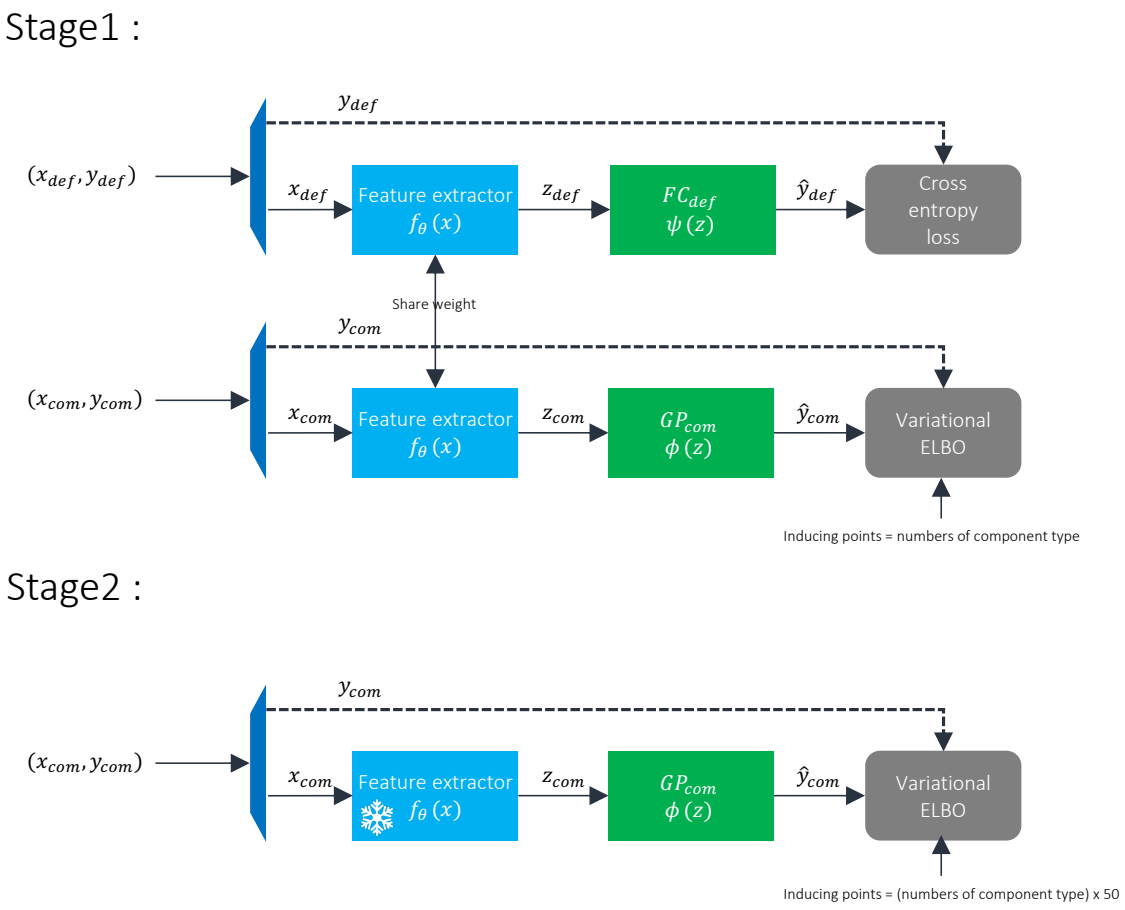


Figure 3.3: training procedure of our method

The training process of our proposed model is divided into two stages, each designed to optimize specific components of the architecture for defect classification and uncertainty estimation.

In the first stage, we train the shared feature extractor and two separate classifiers using distinct datasets tailored for their respective tasks. For the mult-defect-types classification branch, labeled samples (x_{def}, y_{def}), where y_{def} represents defect types, are used to jointly train the feature extractor and the MLP classifier. Simultaneously, for the component classification branch,

labeled samples (x_{com}, y_{com}), where y_{com} corresponds to component types, are used to train the feature extractor and the Gaussian Process (GP) classifier. To ensure the GP classifier focuses on distinguishing component types without being affected by defective samples, we use only ” good” samples for this branch.

This selective training strategy prevents the GP classifier from being confused by defective components, enabling it to maintain higher precision in component classification. Moreover, it ensures that the uncertainty estimates for ”good” components are minimized, further improving the robustness of the GP classifier. By sharing the feature extractor across both branches, the resulting feature vectors possess both the discriminative power required to classify defect types and the capability to estimate the confidence level for any given component type.

In the GP classifier, we initialize the model with the number of inducing points equal to the number of component types. These inducing points act as reference samples, allowing the GP classifier to more accurately estimate the likelihood of each component type. This initial setup establishes a solid foundation for precise component classification.

In the second stage, the parameters of the feature extractor are frozen, and further training is focused exclusively on the GP classifier. Using the same input (x_{com}, y_{com}) as in Stage 1, we increase the number of inducing points by 50 times compared to Stage 1. This significantly enhances the GP classifier's ability to model the likelihood distribution of each component type with greater granularity. By leveraging this refinement strategy, the GP classifier achieves improved performance in its predictions.

We refer to this method as the **Prototypical Gaussian Process Classifier**, as the use of inducing points tailored to the component types enables the classifier to act as a prototype-based model, providing more precise likelihood estimation and confidence calibration for each component type.

Inference Procedure

In the upper branch, the input data is processed through the shared feature extractor and subsequently classified by the Multi-Layer Perceptron (MLP) classifier. The MLP outputs a direct

classification result into one of four defect types: ”good” , ”shift” , ”broke”,or ”short”. This branch focuses on defect type differentiation and provides a detailed categorization of defect samples. The direct classification ensures that defect-specific characteristics are preserved and utilized for high-granularity predictions.

In the lower branch, a more nuanced approach is taken to estimate whether a sample belongs to the categories of good, defective, or indeterminate. This process involves the following key steps:

1. **Probabilities-to-Predict Uncertainty Converter:** The probabilistic output (p_{com}) generated by the Gaussian Process classifier for component type classification is processed to compute the normalized uncertainty (U) using entropy. The entropy-based uncertainty quantifies the confidence of the component type predictions, enabling effective handling of ambiguous samples. The uncertainty measure$U$ is calculated as:

$$U = \frac{1}{\log(K)} \times (-\sum_{i=1}^{K} p_i \cdot \log(p_i)) \tag{3.2}$$

   where K represents the number of component types.

2. **Prediction Uncertainty-to-Defect Converter:** The computed uncertainty (U) is further analyzed to classify the sample into one of the three categories: good, defective, or indeterminate. This conversion is governed by two predefined thresholds, T_j^1 and T_j^2, which are optimized using a Bayesian optimizer to balance the harmonic mean of overkill rate, leakage rate, and indeterminate rate. The classification rules are defined as Equation 3.1

$$Harmonic\,mean = \frac{a+b+c}{\frac{a}{exp(overkill\ rate)} + \frac{b}{exp(leakage\ rate)} + \frac{c}{exp(indeterminate\ rate)}} \tag{3.3}$$

Finally, the results from both the upper and lower branches are integrated in the Prediction Combiner module to yield the final output. The upper branch provides high-resolution defect categorization, while the lower branch ensures accurate handling of uncertain or ambiguous cases. Together, this hybrid inference approach delivers a comprehensive classification result

that leverages both defect-specific and component-specific insights.

Algorithm 3.1 Prediction Combiner $G(\hat{y}^1_{def}, \hat{y}^2_{def})$

Input: the output of upper branch($\hat{y}^1_{def}$), the output of lower branch$\hat{y}^2_{def}$
Output: $\hat{y}_{def}$
if $\hat{y}^1_{def}$ = "good" & $\hat{y}^2_{def}$ = "good" **then**
    $\hat{y}_{def} \leftarrow$ "good"
else if $\hat{y}^1_{def} \neq$ "good" & $\hat{y}^2_{def}$ = "defective" **then** //$\hat{y}^1_{def}$ = "shift"/"broke"/"short"
    $\hat{y}_{def} \leftarrow \hat{y}^1_{def}$
else
    $\hat{y}_{def} \leftarrow$ "indeterminate"
end if
return $\hat{y}_{def}$

3.3 3 Hybrid Experts, each expert incorporating both Single-Defect-Type and Multi-Component-Type Classifier

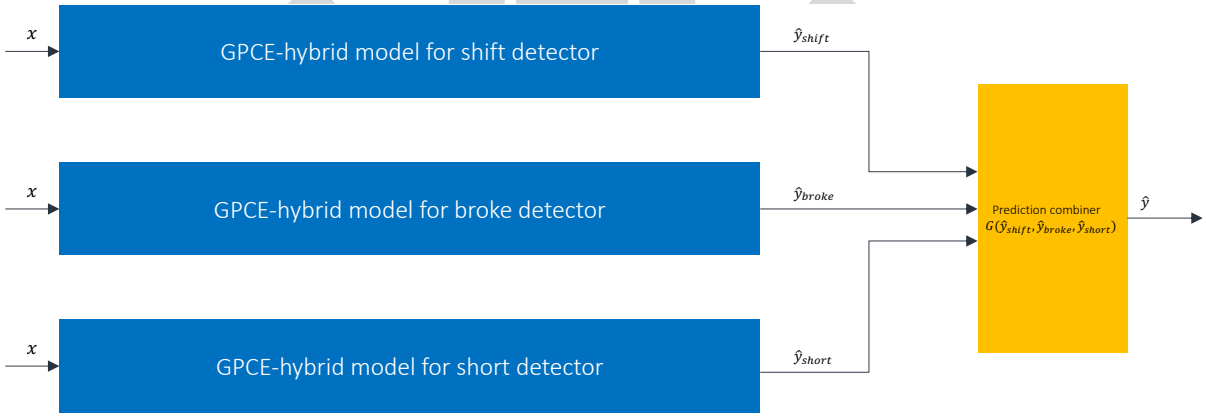


Figure 3.4: Illustration of method2

To further explore the performance of different methods in defect classification, we propose a new approach as a comparative baseline against the previously described method. This approach is based on three independent GPCE Hybrid Models, each dedicated to binary classification for

specific defect types. By decomposing the defect classification problem into multiple binary classification tasks, the model can focus on learning features relevant to individual defect types and achieve comprehensive classification through result integration.

In this method, we categorize defects into three types: ”shift”, ”broke”, and ”short”, and design three independent classifiers accordingly, named the Shift Detector, Broke Detector, and Short Detector, respectively. Each classifier is built on the architecture of a GPCE Hybrid Model, consisting of a shared feature extractor, an MLP classifier, and a Gaussian Process classifier. During training, each classifier is trained individually on its target defect type, determining whether a given sample belongs to the specific defect type, such as ”shift” or ”not shift”. This design enables each classifier to focus on learning features relevant to its target defect, thereby minimizing confusion caused by features associated with other defect types.

During the inference phase, input samples are simultaneously processed by the three independent GPCE Hybrid Models, which individually output classification results for ”shift”, ”broke”, and ”short”. These results include the classification probabilities and final decisions for each defect type. Based on these results, an integrated classification result is generated for the input sample. It is important to note that when a sample is simultaneously classified as belonging to multiple defect types—for example, having features of both ”shift” and ”short”—the sample is labeled as ”indeterminate”. This handling approach not only reduces potential conflicts between classifiers but also effectively prevents misclassification, thereby enhancing the robustness of the system.

The flexibility of this design enables the model to handle complex defect classification scenarios while balancing classification accuracy and uncertainty management. Simultaneously, the model can accurately classify samples with single defect features or determine that a sample is ”good”. This architecture addresses the requirements of multi-type defect classification and provides a reasonable approach to manage uncertain samples.

Training Procedure

In the training process of this approach, we independently train three distinct GPCE Hybrid Models, each targeting a specific defect type: shift, broke, and short. For each model, we utilize a separate dataset with binary labels tailored to the respective classification task. Specifically, the input data and labels for each model are as follows:

For the **Shift Detector**, the dataset consists of samples labeled as true (”shift”) and false (” good,” ”broke,” ”short”), denoted as (x_{shift}, y_{shift}).

For the **Broke Detector**, the dataset includes samples labeled as true (”broke”) and false (” good,” ”shift,” ”short”), denoted as (x_{broke}, y_{broke}).

For the **Short Detector**, the dataset comprises samples labeled as true (”short”) and false (” good,” ”shift,” ”broke”), denoted as (x_{short}, y_{short}).

Each GPCE Hybrid Model shares a common architecture, including a feature extractor, an MLP classifier, and a Gaussian Process classifier. During training, the feature extractor and classifiers are optimized independently for each model, ensuring that the parameters are specifically adapted to their respective binary classification tasks. This independent training strategy enables each model to focus exclusively on distinguishing its target defect type from other categories, minimizing potential confusion caused by overlapping features between defect types.

By isolating the training of each GPCE Hybrid Model, we ensure that the shared feature extractor learns features specific to the target defect type while maintaining the ability to generalize across other categories. This approach enhances the precision of each model’s binary classification capabilities and provides a robust foundation for integrating the results during inference.

Inference Procedure

During the inference phase, input samples are processed independently by the three GPCE Hybrid Models, each specializing in the classification of a specific defect type. The models produce the outputs : $\hat{y}_{shift}, \hat{y}_{broke}, \hat{y}_{short}$

for $defect \in \{shift, broke, short\}$,

$$\hat{y}_{defect} = \begin{cases} \hat{y}^1_{defect}, & \text{if } \hat{y}^1_{defect} = \hat{y}^2_{defect} \\ \text{"indeterminate"}, & \text{if } \hat{y}^1_{defect} \neq \hat{y}^2_{defect} \end{cases} \tag{3.4}$$

After obtaining the outputs from the three independently trained GPCE Hybrid Models ($\hat{y}_{shift}, \hat{y}_{broke}, \hat{short}$), we integrate these outputs to generate the final defect classification result. The integration process combines the binary decisions from the three models to produce a comprehensive categorization of the input sample.

The integration strategy ensures that:

1. **Unique Defect Classification:** If only one of the models classifies the sample as belonging to its target defect type (e.g., only $\hat{y}_{shift} = true$), the sample is assigned to the corresponding defect category (e.g., "shift").

2. **Indeterminate Cases:** If multiple models simultaneously classify the sample as belonging to their respective defect types (e.g., $\hat{y}_{shift} = true$ and $\hat{y}_{short} = true$), the sample is labeled as "indeterminate" to reflect uncertainty and prevent misclassification.

3. **Good Classification:** If none of the models classify the sample as defective(i.e.,$\hat{y}_{shift} = \hat{y}_{broke} = \hat{y}_{short} = false$), the sample is categorized as "good".

Chapter 4

Experiment and Results

4.1 Dataset

This study utilizes a dataset specifically designed for printed circuit board (PCB) inspection to conduct experiments. The dataset contains a total of 136,400 images, with each image labeled with both a component type and a defect type. There are 35 component types represented in the dataset, covering common components found on PCBs, and 6 defect types, which include ”good”, ”missing”, ”shift”, ”stand”, ”broke” and ”short”.

Each image is annotated with two labels: one indicating the component type and the other specifying the corresponding defect type. This dual-labeling design enables the model to learn both component classification features and defect recognition simultaneously, facilitating multi-task learning. It also provides sufficient data to support both component identification and defect classification.

The dataset is highly representative of PCB inspection tasks, making it suitable for evaluating classification models in multi-component and multi-defect scenarios. Additionally, the diversity of the data, which includes images from various component types and multiple defect categories, presents a challenging experimental environment that helps improve the model’s generalization capability.

Table 4.1: Summary for the original dataset.

Number of images	1,364,400
Component types	35
Defect types	6 (good, missing, shift, stand, broken, short)
Label information	Each image is labeled with 1 type of component and 1 type of defect.

Dataset regrouping

In this study, we made a series of adjustments to the dataset to enhance the performance and robustness of the model in defect classification and component classification tasks. Due to the significant visual differences caused by the ”missing” and ”stand” defect types compared to other samples of the same component type, these defects could negatively impact the accuracy of component classification. To address this issue, we treated samples with ”missing” and ” stand” defects as independent component types. This approach reduces the interference of these defects on the learning of component classification features, thereby improving classification robustness.

Figure 4.1: Appearance of different defects in the same component type

In addition, we reorganized the component labels in the dataset by grouping visually similar component types into a single category. This labeling strategy helps the model focus on learning the relationships between components and defects, rather than overfitting to subtle differences between component types. By doing so, we minimize the risk of overfitting in the component classification task and provide the model with stronger generalization capabilities, enabling it to perform consistently across various component and defect combinations.

Lastly, since our study focuses on the classification performance of defect types, we excluded component types that contained no defective samples from the dataset. This ensures that the training and evaluation processes are concentrated on samples relevant to defect classification, avoiding unnecessary interference from non-defective components. This adjustment allows the model’s training and testing to better reflect its actual performance in defect classification tasks.

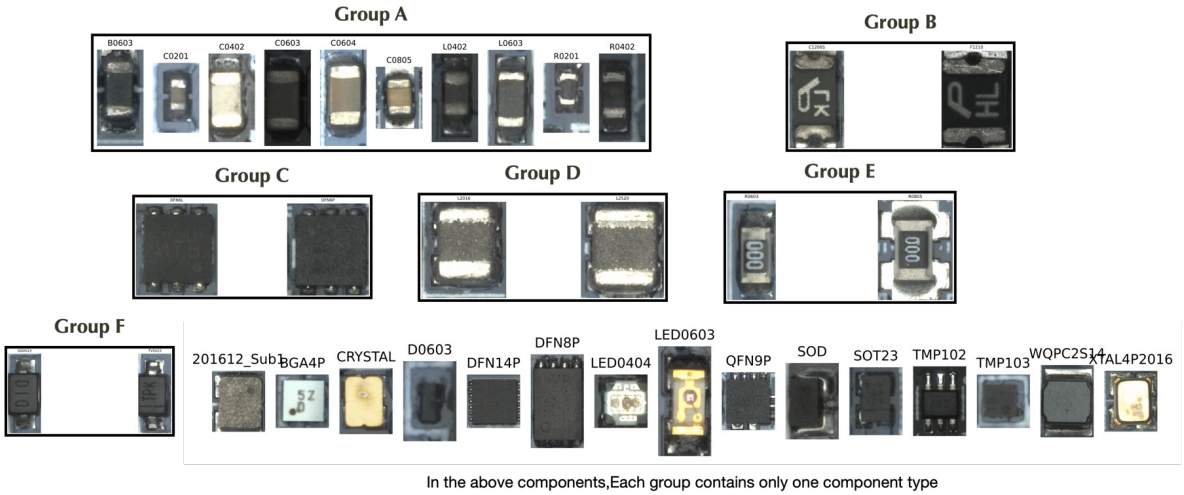


Figure 4.2: Appearance of different component type

Table 4.2: Summary for the our dataset.

Number of images	21991
Component types	10(shown as 4.3)
Defect types	4 (good, shift, broken, short)
Label information	Each image is labeled with 1 type of component and 1 type of defect.

These adjustments make the dataset more suitable for the experimental requirements of this study. They ensure that the model can effectively address both component classification and defect classification tasks while improving the reliability and interpretability of the classification results.

Table 4.3: Number of regrouped component types in our dataset.

component name relabeled	component name	total	good	shift	broke	short
201612_Sub1	['201612_Sub1']	204	204	0	0	0
A	['C0604', 'C0201', 'B0603', 'L0402', 'R0402', 'C0805', 'C0603', 'L0603', 'C0402', 'R0201']	137641	108845	11292	10513	6991
B	['F1210', 'C1206S']	9792	9704	88	0	0
BGA4P	['BGA4P']	3057	2777	178	102	0
CRYSTAL	['CRYSTAL']	6300	6078	76	146	0
D	['L2016', 'L2520']	37458	34294	1245	1479	440
E	['R0805', 'R0603']	1679	1084	449	146	0
F	['TVS523', 'SOD523']	6089	5437	310	342	0
QFN9P	['QFN9P']	773	767	4	2	0
SOT23	['SOT23']	12568	11542	588	438	0
TMP102	['TMP102']	4634	4196	292	146	0

Dataset Split

To ensure the fairness and reliability of model training, validation, and testing, we designed a rigorous data allocation process based on the characteristics of the components, as illustrated in the figure.

First, we divided all components into two categories: **Old Components** and **New Components**, each comprising 5 types of components. The samples from old components are relatively abundant, making them suitable for model training, while new components, with fewer samples, are primarily used to evaluate the model's generalization capability.

1. **Defect Training Set:** The defect training set consists of samples from both old and new components. Specifically, **80% of the old component samples** are included in the defect training set, encompassing both good and defective samples. Additionally, **50% of the good samples from new components** are included in the defect training set to enhance the model's ability to recognize new components. This design ensures that the defect classification model learns features from both old and new components, improving its classification performance across different component types.

2. **Component Training Set:** The component training set is derived from the good samples in the defect training set, consisting of **80% of the good samples from old components**

and **50% of the good samples from new components**. This allocation ensures that the model is not influenced by defective features when performing component classification. By including good samples from new components, the model's generalization capability is strengthened, allowing it to better recognize unseen component types. This setup enables the model to learn both stable old component features and distinguishing features of new components.

3. **Validation Set:** The validation set comprises samples from both old and new components and is used to evaluate the model's performance during training. **10% of the old component samples**, including both good and defective samples, are randomly selected for the validation set. Additionally, **good samples from 2 types new components** is included to assess the model's generalization ability to new components. This balanced design provides comprehensive feedback for model optimization during the training process.

4. **Test Set:** The test set is composed of samples from both old and new components, including the remaining **10% of old component samples (both good and defective)** and **samples from 3 types new components (both good and defective)**. This test set design allows for a thorough evaluation of the model's performance on old components while also assessing its generalization ability on new components.

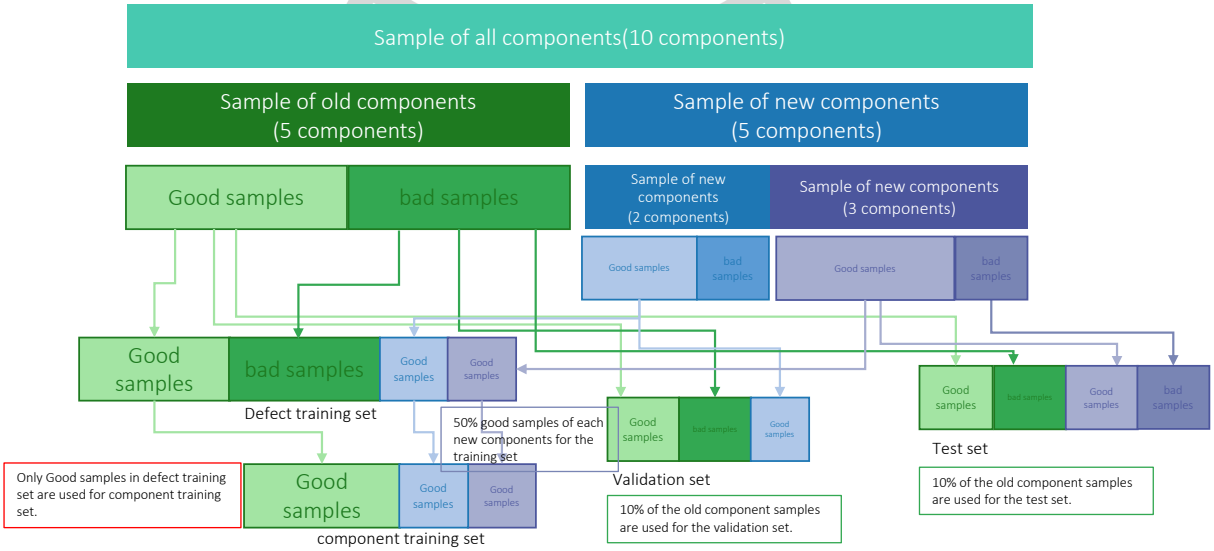


Figure 4.3: Data split between train/valid/test dataset

4.2 Experiment Setup

In this study, we implemented the training and testing processes using the PyTorch framework and **employed MobileNet V3 Large as the backbone network. The model was pre-trained on the ImageNet dataset**, providing robust feature extraction capabilities for our defect classification task. The input image size was set to **3 × 224 × 224 (channels × height × width)** to match the requirements of MobileNet V3.

The model training process consisted of 50 epochs, with a batch size of 128 images per iteration. The initial learning rate was set to $l_{init} = 0.05$, and it was decayed at the 20th, 30th, and 40th epochs to balance the stability and convergence speed of the training process. Two independent stochastic gradient descent (SGD) optimizers were used to update model parameters. One optimizer was dedicated to training the feature extractor f_θ and the MLP classifier α, while the other focused on the feature extractor f_θ and the Gaussian Process classifier β.

To enhance the model's generalization capability, we applied multiple data augmentation techniques to the training dataset. These techniques included random horizontal flipping, random vertical flipping, color jittering, and random grayscale transformations. These augmentations simulated various real-world image variations, thereby improving the model's adaptability to diverse datasets. For the validation and test datasets, no augmentation was applied to ensure the fairness and consistency of the evaluation process.

All experiments were conducted using **a single NVIDIA Tesla V100 GPU**, which provided efficient computational performance and ensured the stability of the experimental results. This configuration supported the reliability of our study, ensuring both the efficiency and reproducibility of the training and testing processes.

Baseline

To evaluate the effectiveness of our proposed method, we designed a baseline model for multi-defect-type classification. This baseline model, as illustrated in the figure, adopts a simple yet efficient two-stage architecture consisting of a feature extractor and a multi-layer perceptron

(MLP) classifier.

The feature extractor in the baseline model utilizes MobileNet V3 Large, which has been pre-trained on the ImageNet dataset. This network effectively extracts both low-level and high-level features from the input images. After processing by the feature extractor, an output feature vector z is generated, which serves as the input to the subsequent classifier.

The classifier is implemented as an MLP with an output layer of 4 dimensions, corresponding to four defect types: good, shift, broke, and short. The classification result $\hat{y}_{def}$ represents the model's prediction for the input image, with possible outcomes being {"good", "shift", "broke", "short"}. Using this baseline model, we compared its classification performance with our proposed method (method1(Section 3.2) and method2(Section 3.3)) on the defect classification task. The simplicity of the baseline design provides a clear and representative benchmark, serving as a foundation for evaluating the improvements in accuracy and robustness achieved by our approach.

Table 4.4: Summary of the experiment setup

GPU	A single NVIDIA Tesla V100 GPU for each experiment.	
Framework	PyTorch	
Backbone	MobileNet V3 Large, pre-trained on ImageNet	
Image size	3 × 224 × 224 (channel × height × width)	
Epoch	30	
Batch size	128	
Learning rate	$lr_{init} = 0.05$, decay the learning rate at 20th, 30th, 40th epoch	
Optimizer	independent SGD optimizers were used for updating model parameter.	
Data Augmentation	Training set: Random horizontal flip, random vertical flip, color jitter, and random gray scale	Validation/test set: No augmentation

4.3 Evaluation Metrics

To comprehensively evaluate the performance of the proposed method, we utilized four key metrics: error rate, overkill rate, leakage rate, and indeterminate rate, each offering insights into different aspects of the model's classification performance.

Error rate

The error rate is a primary metric that measures the overall accuracy of the model by calculating the proportion of incorrect predictions relative to the total number of test samples (excluding those classified as indeterminate). This metric provides a general overview of the model's capability to correctly classify samples across all categories, highlighting its overall reliability in classification tasks.

$$Error\ rate = \frac{(b+c+d+h+i+n)+(f+k+l+p+q+r)}{(a+g+m+s)+(b+c+d+h+i+n)+(f+k+l+p+q+r)} \quad (4.1)$$

Overkill rate

The overkill rate focuses on the misclassification of non-defective samples as defective. This metric is critical in practical applications, as a high overkill rate could result in unnecessary rework or rejection of functional components, leading to increased operational costs. It reflects the model's precision in distinguishing between good samples and those with defects.

$$Overkill\ rate = \frac{(b+c+d)}{(a+g+m+s)+(b+c+d+h+i+n)+(f+k+l+p+q+r)} \quad (4.2)$$

Leakage rate

The leakage rate measures the proportion of defective samples that are incorrectly classified as non-defective. This is particularly significant in quality control scenarios, where undetected defective components could lead to product malfunctions or failures. A low leakage rate indi-

cates the model's robustness in capturing and identifying defective samples accurately.

$$Leakage\ rate = \frac{(f+k+p)}{(a+g+m+s)+(b+c+d+h+i+n)+(f+k+l+p+q+r)} \tag{4.3}$$

Indeterminate rate

The indeterminate rate evaluates the model's ability to manage uncertain predictions by quantifying the proportion of test samples classified as "indeterminate." This metric is especially important for ensuring system reliability in high-stakes scenarios where misclassification could have severe consequences. By categorizing uncertain predictions as indeterminate, the model effectively mitigates the risk of erroneous decisions, contributing to overall system robustness.

$$Indeterminate\ rate = \frac{(e+j+o+t)}{(a+g+m+s)+(b+c+d+h+i+n)+(f+k+l+p+q+r)+(e+j+o+t)} \tag{4.4}$$

Table 4.5: Evaluation Metrics.

actually \ predicted	good	shift	broke	stand	indeterminate
good	a	b	c	d	e
shift	f	g	h	i	j
broke	k	l	m	n	o
short	p	q	r	s	t

4.4 Experiment Result

To comprehensively demonstrate the performance of the proposed method, we adopted a variety of visualization and quantitative evaluation approaches. These methods include learning curves, t-SNE visualizations, confusion matrices, and a series of well-defined evaluation metrics, which collectively provide an in-depth analysis of the model's performance from different perspectives.

In the subsequent presentation of results, we define the two proposed approaches for clarity and comparison. The first approach, referred to as Method 1(Section 3.2), is a Hybrid Expert that incorporates both multi-defect-type and multi-component-type classification capabilities within a single framework. This method is designed to simultaneously handle multiple defect types and component types, leveraging a unified structure for comprehensive defect and component classification.

The second approach, referred to as Method 2(Section 3.3), involves the use of three independent Hybrid Experts, where each expert is specifically tailored for a single defect type while also incorporating multi-component-type classification. By dividing the defect classification task into three separate binary classifiers (one for each defect type), this method allows for more focused learning on individual defect types while maintaining component classification capabilities.

These two methods represent distinct strategies for addressing the challenges of defect and component classification, and their performance will be compared in terms of classification accuracy, robustness, and uncertainty management in the experimental results.

4.4.1 Quantitative Evaluation Metrics

To provide additional quantitative assessments, we defined a set of evaluation metrics, including error rate, overkill rate, leakage rate, and indeterminate rate. The formulas for these metrics are shown in Section 4.3

Table 4.6: (All components) Performance of Baseline,Method1,Method2

	Error rate (%)	Overkill rate (%)	Leakage rate (%)	Indeterminate rate (%)
Baseline	3.898±0.509	1.498±0.374	1.868±0.545	-
Method1-u	2.303±0.736	0.194±0.150	2.039±0.559	8.989±1.581
Method1	2.293±0.725	0.192±0.153	2.077±0.565	10.732±1.706
Method2-u	6.377±0.410	0.034±0.008	6.313±0.393	11.472±0.132
Method2	6.400±0.432	0.033±0.007	6.340±0.419	12.126±0.550

Table 4.7: (Old components) Performance of Baseline,Method1,Method2

	Error rate (%)	Overkill rate (%)	Leakage rate (%)	Indeterminate rate (%)
Baseline	1.955±0.492	1.939±0.495	0.004±0.003	-
Method1-u	0.479±0.362	0.463±0.364	0.002±0.003	7.553±2.403
Method1	0.493±0.377	0.477±0.379	0.002±0.003	10.323±2.502
Method2-u	0.314±0.112	0.087±0.015	0.153±0.108	19.044±2.357
Method2	0.414±0.100	0.087±0.015	0.153±0.099	19.362±2.218

Table 4.8: (New components) Performance of Baseline,Method1,Method2

	Error rate (%)	Overkill rate (%)	Leakage rate (%)	Indeterminate rate (%)
Baseline	5.195±0.440	0.382±0.219	3.084±0.881	-
Method1-u	3.536±0.987	0.012±0.012	2.637±0.394	9.980±1.938
Method1	3.478±0.931	0.006±0.005	2.703±0.362	11.093±2.789
Method2-u	9.840±0.855	0.005±0.006	9.833±0.850	6.482±1.536
Method2	7.379±1.052	0.002±0.002	9.902±0.916	7.379±1.052

4.4.2 Learning Curve

The learning curve is used to illustrate the model's learning trend during training by plotting the training loss and validation loss against the number of epochs. This curve helps to analyze the convergence and stability of the model and visually indicates whether the model is experiencing

overfitting or underfitting, providing valuable guidance for parameter tuning and architecture optimization.

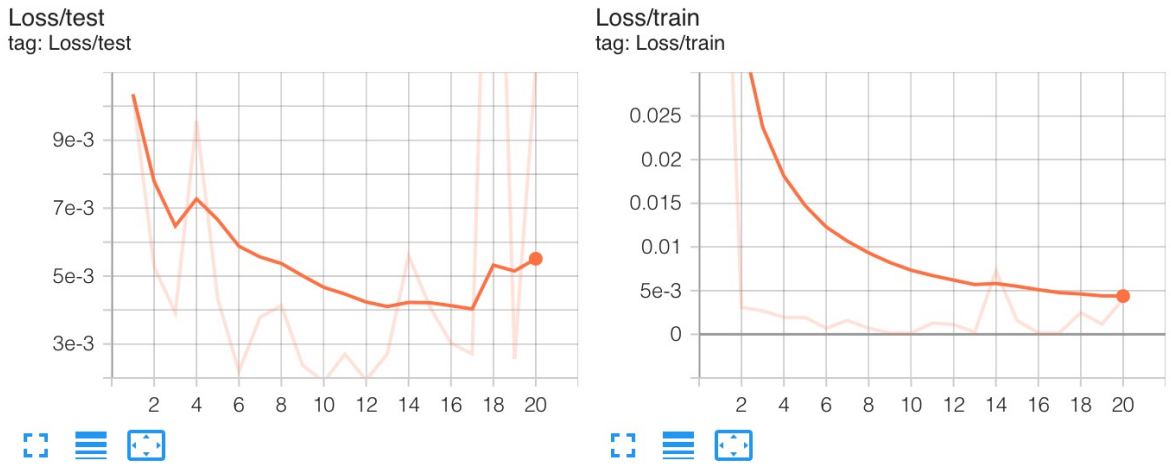


Figure 4.4: Learning curve of Baseline

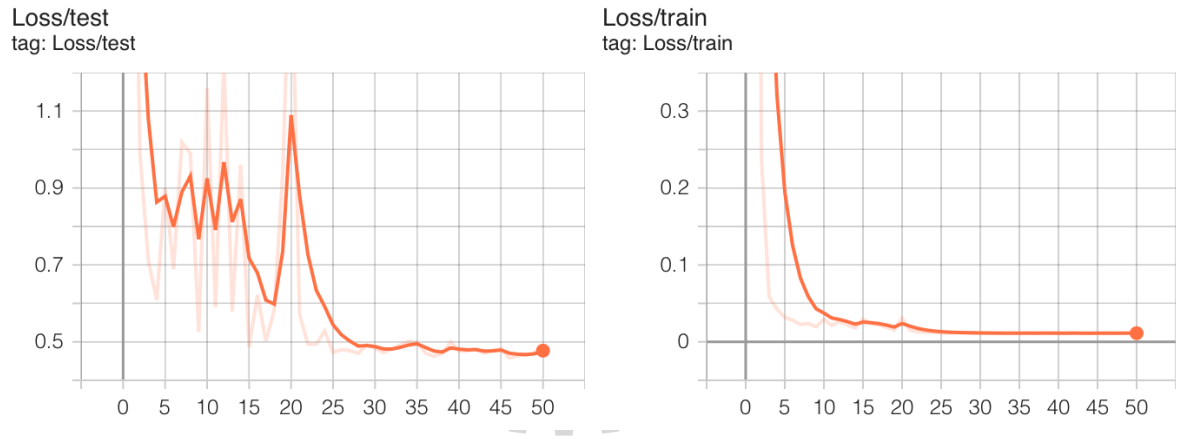
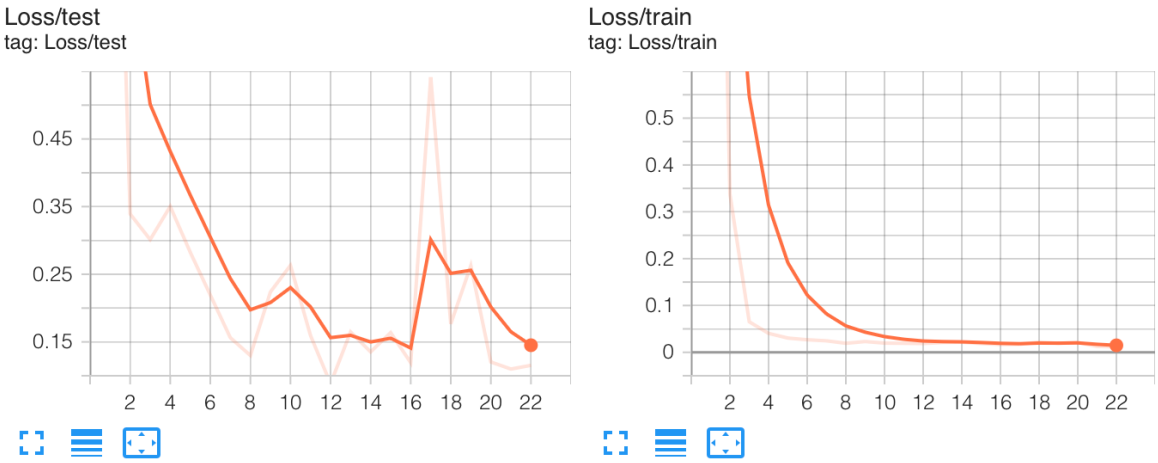
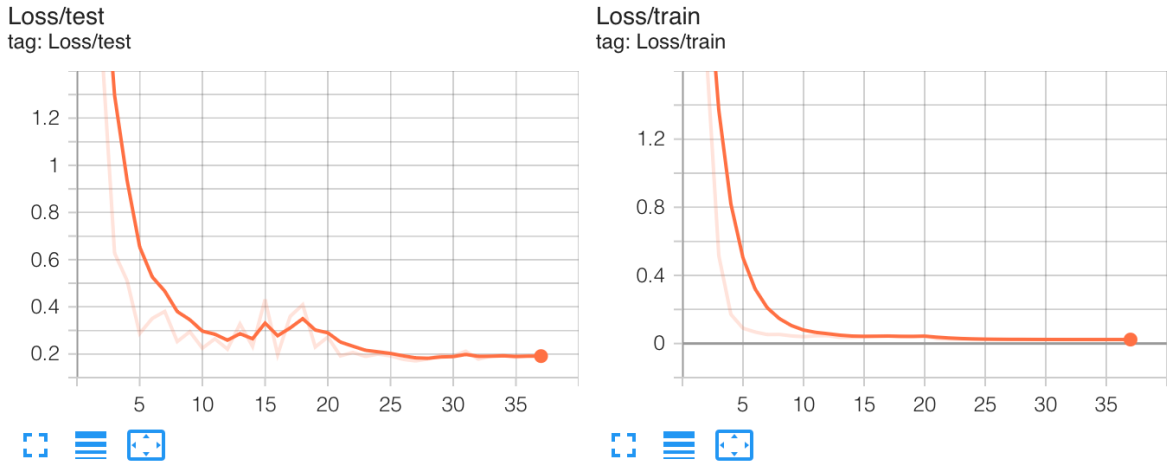


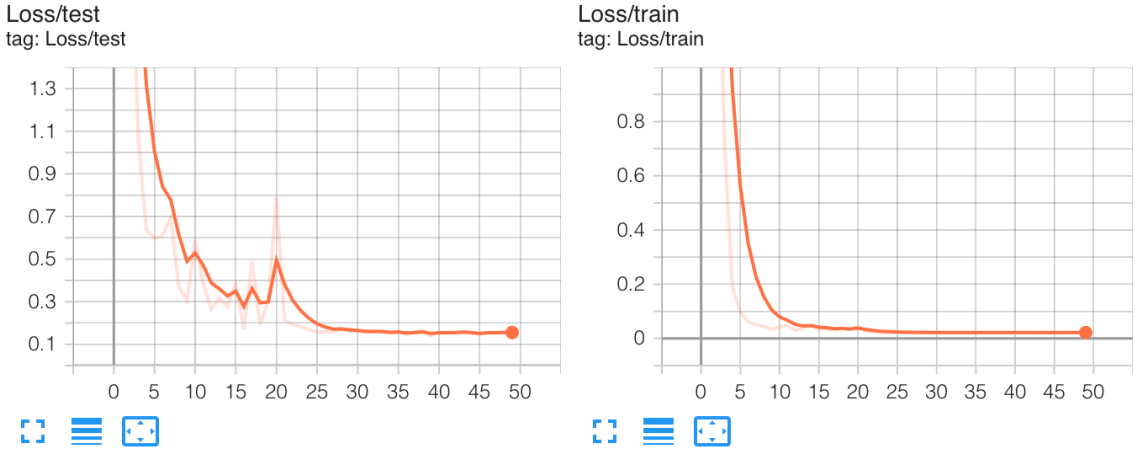
Figure 4.5: Learning curve of Method1



(a) shift detector



(b) broke detector



(c) short detector

Figure 4.6: Learning curve of Method2.

4.4.3 t-SNE Visualization

To showcase the distribution of the feature vectors extracted by the model, we employed t-SNE (t-Distributed Stochastic Neighbor Embedding). By reducing the dimensionality of high-dimensional feature vectors to two dimensions, t-SNE provides an intuitive visualization of the separability between different component or defect categories. In the t-SNE plots, we highlight the distributions of samples from various categories, demonstrating the model's ability to distinguish between defect and component types and validating the effectiveness of the proposed feature extraction method.

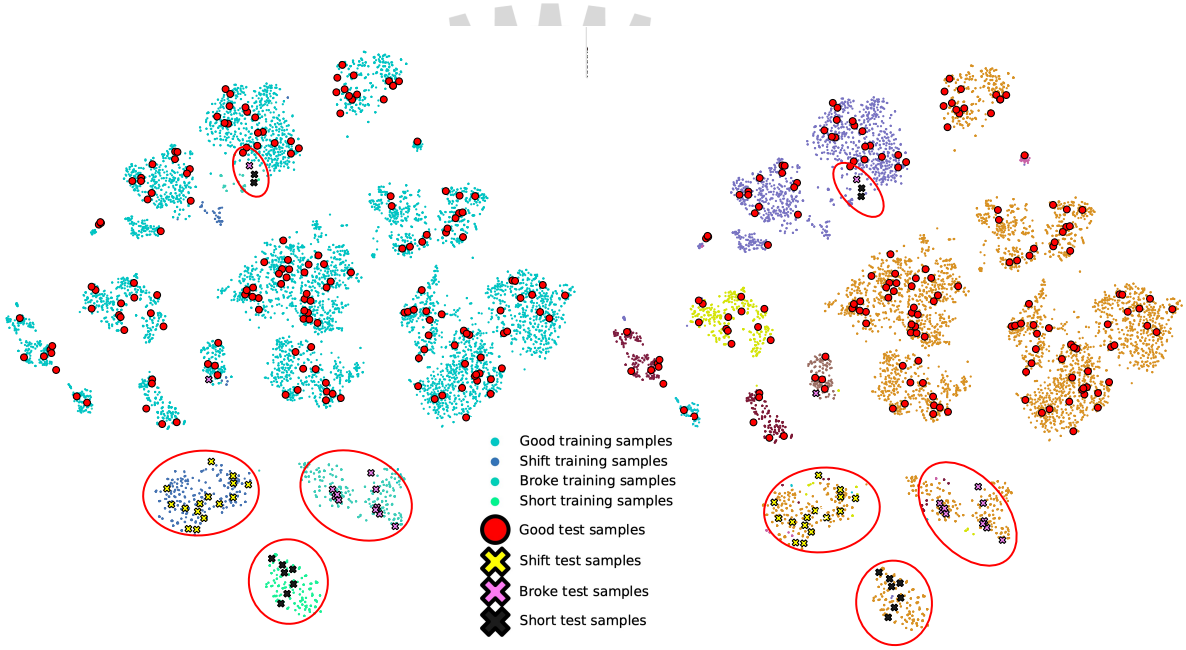


Figure 4.7: t-sne of Baseline

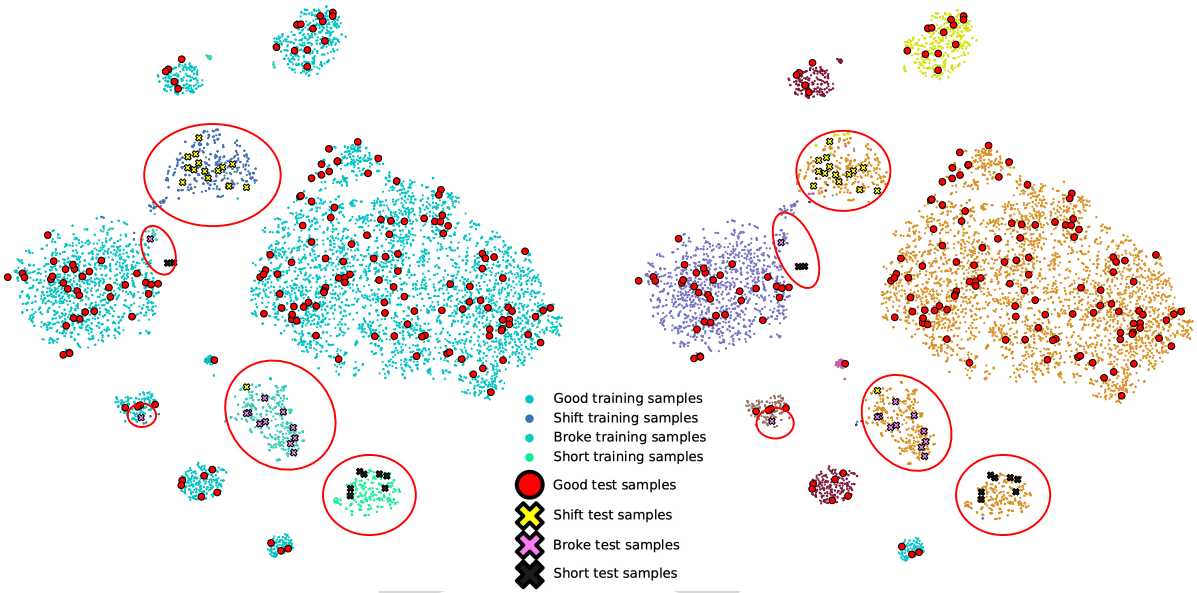
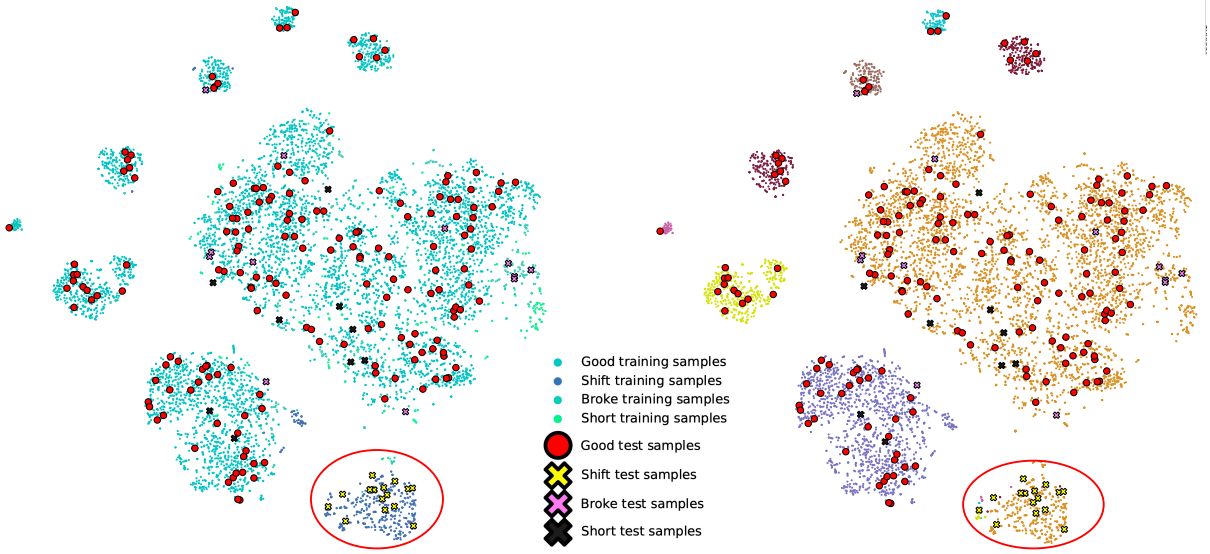
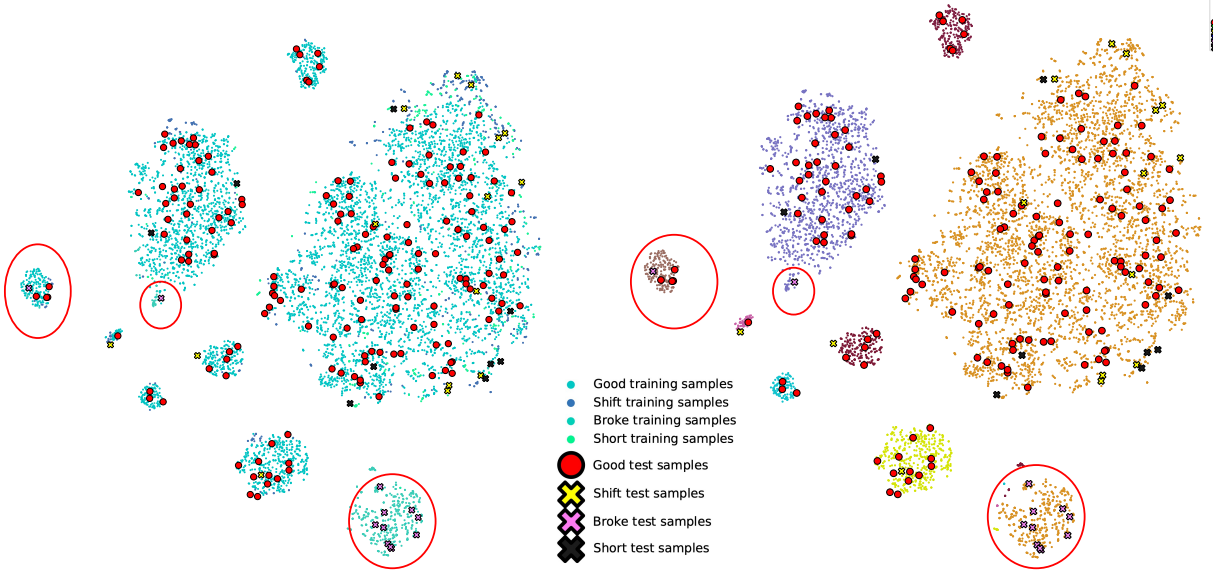


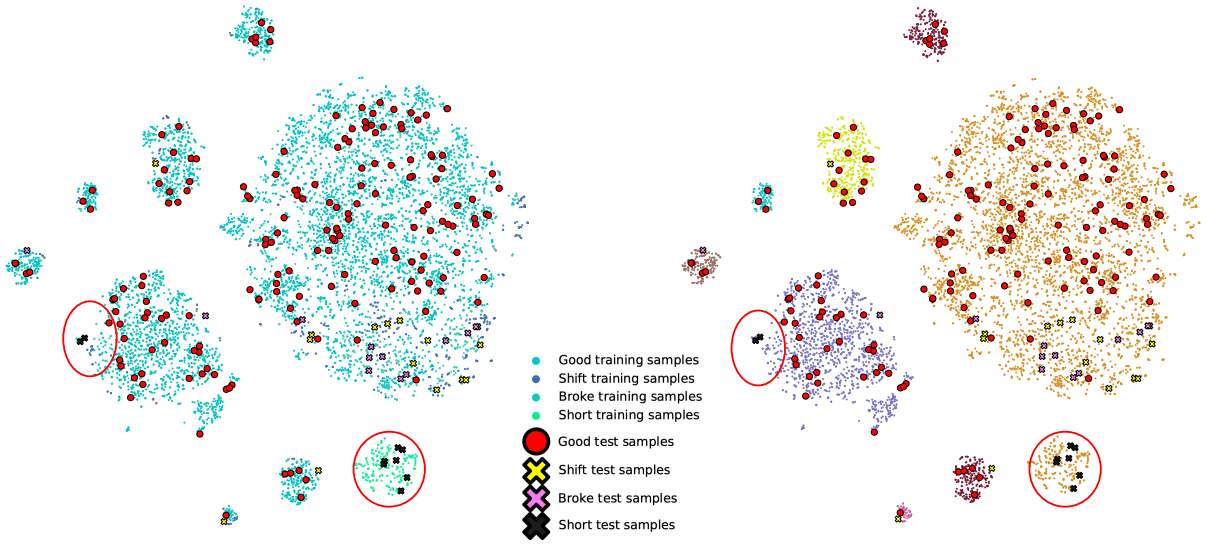
Figure 4.8: t-sne of Method1



(a) shift detector



(b) broke detectot



(c) short detector

Figure 4.9: t-sne of Method2.

4.4.4 Confusion Matrix

The confusion matrix is a widely used tool for analyzing classification results, quantifying the accuracy and misclassification rates for each class. We generated confusion matrices to compare the classification results of the baseline model with our proposed method. Each element in the matrix represents the correspondence between actual and predicted categories. By examining the confusion matrix, we further explored the model's classification performance on specific defect types, especially patterns and causes of misclassification.

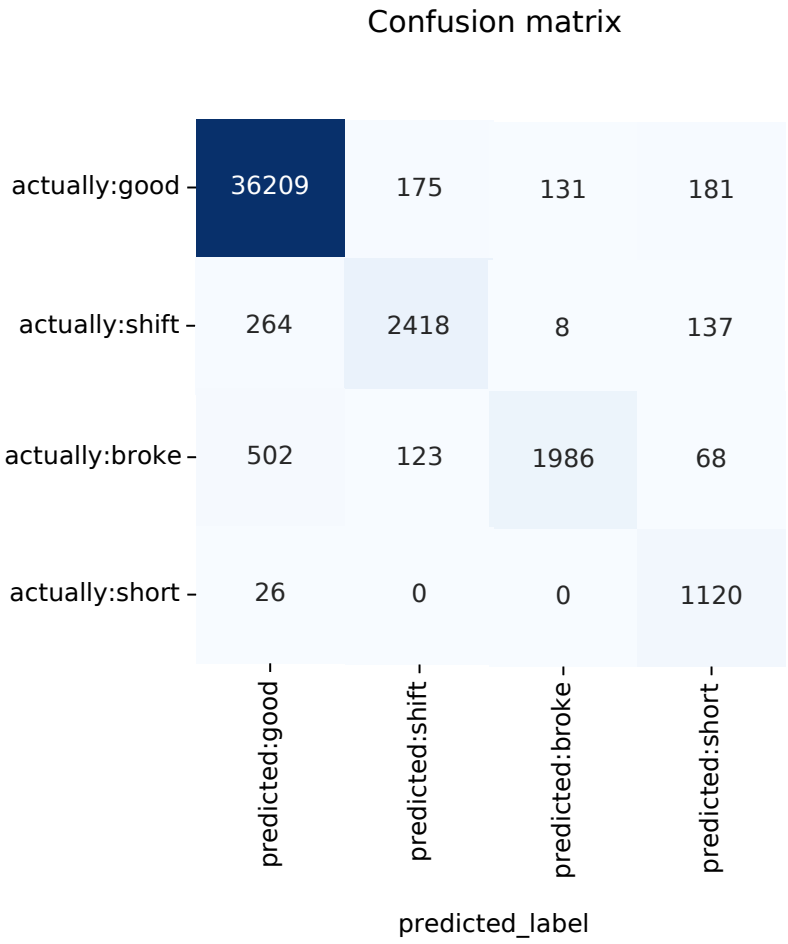


Figure 4.10: confusion matrix of Baseline

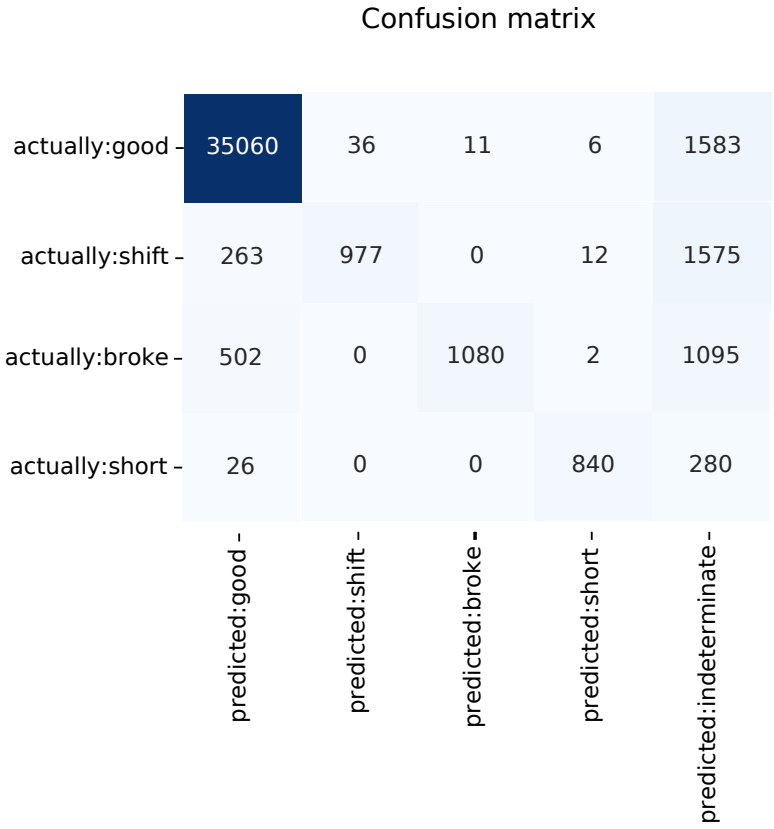


Figure 4.11: confusion matrix of Method1

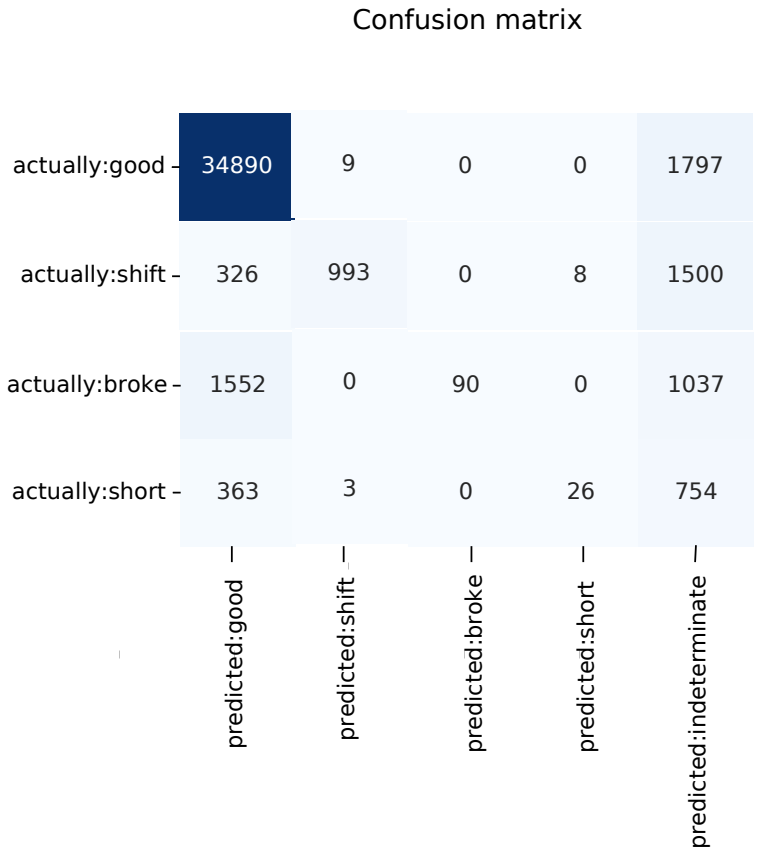


Figure 4.12: confusion matrix of Method2

Chapter 5

Conclusions and Future Works

In this study, we compared the performance of a baseline model, Method 1 (a Hybrid Expert incorporating both Multi-Defect-Type and Multi-Component-Type Classifier), and Method 2 (three Hybrid Experts, each incorporating both Single-Defect-Type and Multi-Component-Type Classifier). Among these, Method 1 demonstrated the best overall performance, significantly outperforming both the baseline and Method 2. Conversely, Method 2 exhibited the worst performance, highlighting the challenges inherent in its design.

Upon analysis, two primary reasons for the performance gap were identified. First, the dichotomy of categorizing data into specific and non-specific defects in Method 2 may have caused confusion for the Gaussian Process classifier. This issue arose because the Gaussian Process classifier encountered defect types beyond its specific target defect, potentially leading to less reliable predictions. Second, the significant increase in data imbalance ratios, inherent to the structure of Method 2, adversely impacted the Multi-Layer Perceptron (MLP) classifiers. This imbalance likely contributed to a degradation in the overall classification performance.

These findings emphasize the importance of designing models that can effectively balance defect classification complexity with robustness to data imbalance. The superior performance of Method 1 underscores the value of an integrated multi-defect and multi-component classification approach in achieving accurate and reliable results.

References

[1] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[2] H. Han, W.-Y. Wang, and B. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," *Proceedings of the International Conference on Intelligent Computing*, vol. 3644, pp. 878–887, 2005.

[3] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[4] C.-L. Zhang, S. Zhang, and Y. Xu, "Deep learning-based surface defect detection for steel plates: A review," *Artificial Intelligence Review*, vol. 52, no. 4, pp. 341–376, 2019.

[5] J. Huang, Y. Wu, and W. Zhang, "Attention mechanism-based convolutional neural network for surface defect detection in industrial environments," *Expert Systems with Applications*, vol. 176, p. 115982, 2021.

[6] H. Li and Y. Li, "Transfer learning-based defect classification for industrial manufacturing using pre-trained cnn models," *Engineering Applications of Artificial Intelligence*, vol. 92, p. 103736, 2020.

[7] B. Settles, *Active learning*. Morgan Claypool Publishers, 2012, vol. 18.

[8] Z. Zhang, F. Zhao, and L. Chen, "Hybrid deep learning models for defect classification," *Computers & Industrial Engineering*, vol. 125, pp. 604–616, 2018.

[9] J. Wu and M. Zhang, "Integrating recurrent neural networks and statistical models for defect classification in time-series datasets," *Expert Systems with Applications*, vol. 159, p. 113439, 2020.

[10] J. Sun and H. Li, “Multi-task learning-based hybrid models for defect and component classification,” *IEEE Access*, vol. 9, p. 3107072, 2021.

[11] J. Chen and T. Wang, “Ensemble hybrid deep learning models for complex defect classification,” *Engineering Applications of Artificial Intelligence*, vol. 105, p. 105572, 2022.

[12] W. Wang and J. Zhang, “Autoencoder and random forest-based hybrid models for defect classification in imbalanced datasets,” *Expert Systems with Applications*, vol. 174, p. 114698, 2021.

[13] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” *Proceedings of the International Conference on Machine Learning*, vol. 48, pp. 1050–1059, 2016.

[14] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” *Advances in Neural Information Processing Systems*, vol. 30, pp. 6405–6416, 2017.

[15] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT Press Cambridge, MA, 2006, vol. 1.

[16] L. Wu, W. Zhang, and F. Li, “Industrial defect detection using gaussian processes: Probabilistic predictions and uncertainty measures,” *IEEE Transactions on Industrial Electronics*, vol. 68, no. 5, pp. 4632–4643, 2021.

[17] A. Kendall and Y. Gal, “What uncertainties do we need in bayesian deep learning for computer vision?” *Advances in Neural Information Processing Systems*, vol. 30, pp. 5574–5584, 2017.

[18] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, “Weight uncertainty in neural networks,” *Proceedings of the International Conference on Machine Learning*, vol. 37, pp. 1613–1622, 2015.

[19] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[20] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” *Advances in Neural Information Processing Systems*, vol. 25, pp. 2960–2968, 2012.

[21] D. Maclaurin, D. Duvenaud, and R. P. Adams, “Gradient-based hyperparameter optimization through reversible learning,” *Proceedings of the International Conference on Machine Learning*, vol. 37, pp. 2113–2122, 2015.

[22] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4780–4789, 2019.

[23] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” *Proceedings of the International Conference on Learning Representations*, 2017.