A COMPUTER VISION MODEL FOR INDUSTRIAL PRODUCT SURFACE INSPECTION USING UNSUPERVISED LEARNING AND FEW-SHOT LEARNING

SHUAI LYU

PhD

The Hong Kong Polytechnic University

2025

The Hong Kong Polytechnic University

School of Fashion and Textiles

A Computer Vision Model for Industrial Product Surface Inspection using Unsupervised Learning and Few-shot Learning

Shuai LYU

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

August 2024

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Shuai LYU
_____ (Name of student)

Abstract

The advancement of AI has significantly accelerated CV research in industrial manufacturing, focusing on using AI-based models to detect and classify surface defects in product inspection. However, existing models often underperform in practice due to neglecting defect characteristics. This study addresses three key issues: underutilization of large-scale normal samples, challenges in leveraging few defective samples, and ensuring robust detection against product/inspection standard changes. While industrial settings have abundant normal samples, effectively using them for training remains difficult. Humans can detect/classify defects with minimal examples, but deep learning requires substantial labeled data, which is often unfeasible. Lastly, current models struggle to generalize to new products/standards due to industrial products' vast diversity and irregularity.

First, the unsupervised Reducing Biases (REB) is proposed for industrial anomaly detection representation. This model learns from normal images to detect product surface defects. A self-supervised task fine-tunes the pre-trained model via the DefectMaker strategy, ensuring diverse synthetic defects. Additionally, the local-density k-nearest neighbors (LDKNN) method addresses normal image patterns/distributions, reduces feature space density bias, and enables effective anomaly detection.

Second, this research proposes a novel mulit-view region context (MVREC) framework for few-shot defect multi-classification (FSDMC), focusing on generalization and contextual feature extraction. MVREC enhances defect classification by integrating pre-trained AlphaCLIP for general feature extraction and using a region-context framework with multi-view context augmentation. The framework also includes Few-shot Zip-Adapter(-F) classifiers to cache support set features for efficient few-shot classification. To validate MVREC, this study introduces MVTec-FS, a new FSDMC benchmark with instance-level mask annotations for 46 defect categories, and demonstrates its performance via comprehensive experiments on multiple datasets.

Third, addressing fabric inspection challenges in the dynamic clothing industry—where new fabric types and inspection criteria emerge—this research proposes a Continual Learning-based Fabric Inspection Model (CLFIM). Traditional models fail to adapt to unseen patterns or defect categories due to pattern/criteria shifts. CLFIM, trained on specific fabrics, learns pattern and inspection criteria contexts to adapt to new tasks. Experiments on three complex-patterned fabrics with varying criteria show that the YOLOV8-based CLFIM effectively handles evolving inspection challenges.

Overall, this research introduces innovative approaches—REB, MVREC, and CLFIM—to address critical challenges: underutilization of normal/defective samples and detection difficulties for altered products/inspection criteria. These models enhance defect detection/classification across fabric types and industrial products. Incorporating self-supervised, few-shot, and continual learning expands defect detection research scope.

Publications arising from the thesis

[1] Shuai Lyu, Dongmei Mo, Wai Keung Wong, "REB: Reducing biases in representation for industrial anomaly detection," *Knowledge-Based Systems*, vol. 290, pp. 111563, 2024. DOI: https://doi.org/10.1016/j.knosys.2024.111563.

[2] Lyu S, Zhang R, Ma Z, et al. MVREC: A General Few-shot Defect Classification Model Using Multi-View Region-Context[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2025, 39(6): 5937-5945.

Acknowledgements

I would like to extend my deepest gratitude to my supervisor, Prof. Wai Keung Wong, for his expert guidance, invaluable insights, and unwavering support throughout my doctoral journey. His expertise and dedication have been instrumental in shaping this research and inspiring me to pursue excellence. I am also profoundly thankful to the faculty and staff of the School of Fashion and Textiles for fostering a supportive academic environment and providing the resources crucial to my research. My sincere thanks go to my colleagues and fellow researchers, whose support, collaborative efforts, and insightful discussions have been a continuous source of inspiration and motivation. And I am deeply grateful to my family for their enduring love, encouragement, and understanding, which have given me the strength to overcome challenges. Their sacrifices and steadfast support have been the foundation of my perseverance in this endeavor. Lastly, I am deeply grateful to my girlfriend GULIJIANNAITI for her five years of unwavering companionship, which has provided me with immense emotional support and strength, especially during the most challenging moments of my doctoral studies.

Table of Contents

Chapter 1

Introduction

1.1 Background

In manufacturing Industry, productivity and product quality are fundamental to a company's growth and competitiveness. Consequently, defect detection on industrial product surfaces has garnered significant attention from researchers across various industrial sectors, including the textile, steel, automotive industries, intelligent manufacturing, semiconductors, and polymer material processing. Surface quality is critical for products because surface defects can greatly affect their appearance, functionality, performance, yield, management, and marketability. The limitations of current product surface inspection methods are a key factor constraining product quality. As labor costs rise, demand for production capacity expands, relevant policies provide support, and numerous industrial frameworks are established, new methods such as machine vision-based intelligent defect detection are rapidly emerging. In industrial production, manual measurement and judgment are prone to errors, visual fatigue, and inconsistencies among operators. Despite these challenges, ineffective and costly human visual inspection remains the predominant approach. To further advance the automation of product defect detection, the adoption of computer vision (CV) and deep learning (DL) methods has emerged as a key development direction. These methods simulate human inspection processes to enhance efficiency and accuracy in detecting product defects. Consequently, the study

of CV methods for product defect detection holds significant practical relevance, and research in this area is of great theoretical importance and practical value for achieving intelligent manufacturing.

Industrial surface inspection is typically conducted using image processing within computer vision-based (CV) techniques. CV-based defect detection, with its long history of development, identifies the presence of defects in an image and determines their location. This work classifies CV-based defect detection into two main categories: traditional CV-based methods and deep learning-based (DL) methods.

1.1.1 Traditional Computer Vision-based Defect Detection

Traditional CV-based algorithms have played a significant role in research, especially before the widespread adoption of convolutional neural networks (CNNs). These traditional algorithms aim to construct templates or features for detecting defects, and their performance heavily depends on how accurately they can model these features. Numerous methods have been proposed to address this challenge across various scenarios. Popular feature-extraction techniques include the local binary pattern [1], co-occurrence matrix [2], Gabor filter [3], and histogram [4]. However, traditional algorithms often struggle with generalization and effectiveness, as they are typically designed for specific scenarios.

1.1.2 Deep Learning-based Defect Detection

Designing generic feature representations for defect detection tasks presents a significant challenge. Deep learning (DL)-based defect detection, which falls under the category of representation learning, can automatically learn defect features and has recently shown greater potential, becoming one of the most prominent topics in AI. Today, several DL-based models [5, 6] have significantly outperformed traditional methods in defect detection due to their ability to automatically learn task-relevant representations. These DL models, in which all the samples used for training are labeled, are referred to as fully supervised models. However, training

a convolutional neural network (CNN) requires a large number of annotated samples, particularly defective ones. The process of collecting and annotating these samples is time-consuming, which significantly limits their industrial application. To address the high cost of labeling, many weakly supervised models [7, 8, 9] and unsupervised models [10, 11, 10] have been developed, and these have also become hot research topics. Weakly supervised models rely on imprecise supervision, which reduces costs. Liu et al. [7] proposed an effective weakly supervised model for fabric defect detection, incorporating attention mechanisms to emphasize defective regions. Zhang et al. [8] suggested training a weakly supervised model with image-level labels while performing both image classification and defect localization. Boi et al. [9] explored using a mixture of image-level and pixel-level labels to enhance image classification performance. However, weakly supervised learning methods may face unstable training due to incomplete supervision, resulting in unsatisfactory performance. Unsupervised models, such as autoencoders, estimate defects by calculating the residual between an image and its reconstruction. These models assume that only non-defective samples are used during training to eliminate the need for data labeling, with the expectation that reconstruction errors will be higher for unseen defective images. Mei et al. [10] proposed a multi-scale convolutional denoising autoencoder network for unsupervised fabric defect detection. Tsai and Jen [11] also employed a convolutional autoencoder (CAE) to detect surface defects, significantly reducing development costs. Niu et al. [12] trained a generative adversarial network to measure the distribution distance between test samples and non-defective samples. However, the results are not always as expected, as CNNs may utilize their powerful learning abilities to replicate inputs, including defective regions.

1.2 Research Gap

To summarise, previous works cannot be applied properly to actual industrial scenarios due to ignoring that a large number of normal samples with a small number of defective samples is most often encountered in industrial scenarios. The

research gap is as follows:

(1) Underutilization of Large-Scale Normal Samples

These data-driven DL-based methods rely on large-scale defect datasets with high-quality annotations to alleviate the overfitting problem. However, it is difficult to collect a big-scale fabric defect dataset due to the low probability of defect occurrence. Luckily, Large-Scale normal (non-defective) samples are easy to collect in industrial scenarios. How to make the best of the large-scale normal samples to train a defect detection model is a crucial research question.

(2) Underutilization of a small number of defective samples

In practice, a small number of defective samples of different classes is available in addition to a large number of normal samples. Even if only a few defect samples of each class are provided, humans can learn characteristics and do detection and classification with a high degree of accuracy. Unfortunately, deep learning algorithms rely on as much labeled data as possible to obtain great generalization, which is not in line with the actual situation. In particular, defect classification tasks require a very large number of labeled samples to cope with the intra-class diversity of the defects. Thus, it is a practical research problem to use a small number of defective samples to develop a model for defect classification properly.

(3) The uncertain performance when products and inspection standards are altered.

The uncertain performance when products and inspection standards are altered. It is challenging to build an annotated dataset and train a model that covers all products and potential defects due to the extensive diversity of industrial products and the irregularity of defects. Most current models are typically trained to detect specific defects in a controlled set of products with predefined inspection criteria. However, when these products or inspection standards are altered, the performance of the model becomes uncertain. The model may struggle to adapt to new types of defects or product variations it has not encountered during training. This limitation poses significant challenges in maintaining consistent accuracy and reliability in

real-world applications.

1.3 Research Objectives

This research seeks to design an intelligent model for the quality control of industrial products. The primary objectives of this study are outlined as follows:

a) To develop a new intelligent vision model for general industrial image defect detection and classification. The defect detection model learns from a large number of normal images (without defects) and can detect defects subsequently. On the other hand, the defect classification model can identify the defect categories.

b) To train a defect detection model by using only normal samples for handling the lack of defect data. The model learns the normality representation from normal samples to handle unseen defects and achieve better detection precision and recall.

c) To develop a defect classification model based on the few-shot learning to make the best of the limited defects and achieve competitive classification accuracy.

d) To apply the model in the textile industry and handle fabric inspection problems on common woven fabrics, including plain, stripe and print fabric.

1.4 Research Methodology

In this study, a comprehensive Computer Vision Model for Industrial Product Surface Inspection is proposed, specifically focusing on fabrics in the textile and apparel industry. The research is organized into four key phases.

First, this study constructed a dataset by combining public and self-collected sources, including the MVTec AD dataset, Visa dataset, and the AIFIX fabric dataset, to evaluate the models. The MVTec AD dataset, a widely recognized benchmark for industrial anomaly detection, along with the AIFIX fabric dataset and the self-collected fabric dataset, were used to assess the performance of the few-shot classification and context-based fabric inspection models.

The second phase involves industrial anomaly detection, where this study utilizes pre-trained models to detect anomalies in industrial product images, framing it as a one-class classification problem that addresses domain bias and normality distribution.

In the third phase, this study addresses few-shot defect multi-classification at the instance level using techniques such as multi-view augmentation, region context representation, fine-tuning, and adapter-based models.

The final phase introduces a continual learning-based fabric inspection model designed to tackle the challenges of pattern shift and inspection criterion shift in fabric inspection tasks. The model undergoes initial pre-training on a plain color fabric dataset, followed by continual learning using support samples to detect defects in unfamiliar fabrics. A thorough examination of the methodology is detailed in Chapter 3.

1.5 Research Significance

This study proposes a comprehensive model with corresponding algorithms to address the challenges of defect detection and classification in industrial product surface images, particularly fabric images. The model is capable of highlighting and displaying defect locations on fabrics, facilitating quality inspection in the industrial manufacturing industry. Additionally, the model can categorize defects, aiding in tracking the causes of these defects. specifically, this study contributes to the field of computer vision in the following aspects:

(1) Enhancement of Methodology for Defect Detection.

This study introduces an algorithm designed to enhance defect detection performance. The algorithm maximizes the utilization of normal samples to train a defect detection model, thereby improving the detection of unseen defects. Furthermore, the algorithm effectively leverages pre-trained models to analyze surface images and accurately localize defects.

(2) Advancement of Methodology for Defect Classification.

This study offers new perspectives to broaden the research methodology for defect classification. It advocates for the direct application of general feature extraction from pre-trained models, combined with few-shot defect samples, to train a defect classification model. This approach addresses the gap in methodologies for defect classification in quality inspection.

(3) Improvement of Performance for Fabric Inspection Model.

This study proposes a fabric inspection model utilizing continual learning, specifically designed for the textile industry. Pre-trained feature representations and memory bank techniques are introduced to quickly adapt to new detection tasks involving different fabric types and defect categories. This approach aims to enhance the performance of defect category classification in the textile industry.

1.6 Outline of the Thesis

Chapter 1 introduces the research background of industrial quality control. This chapter highlights the research gap and describes the research objectives and significance of this study. Chapter 2 rviews the research works on the CV field. Chapter 3 presents the research methodology for solving the defect detection and classification problems, including the defect dataset, the metrics, and the proposed method. Chapter 4 introduces an unsupervised anomaly detection algorithm for industrial product quality control. Chapter 5 introduces a defect multi-classification model using few-shot learning. Chapter 6 introduces a Continual Learning-based Fabric Inspection Model to adapt to unseen defects and fabric types. Chapter 7 summarises the accomplished work and gives plans for future research.

Chapter 2

Literature Review

Deep learning is increasingly used in industrial product surface inspection. This chapter first reviews the use of DL in industrial product surface inspection. In addition, the paradigm of anomaly detection using normal images, few-shot defect multi-classification, and continual learning-based fabric inspection are discussed.

2.1 Deep Learning-based Models for Defect Detection

This work examines deep learning-based algorithms for defect detection from the perspective of learning methodologies, including supervised and unsupervised learning. Supervised models require that all images be labeled. Based on their network structure, these models can be categorized into three types: defect classification models, object detection models, and segmentation models. Defect classification models address the "what" question, i.e., determining whether an image contains defects and identifying the category of those defects. These models can be further divided into binary and multi-class classifications. The binary classification models aim to predict the presence of defects in an image, while the multi-class models seek to identify the specific class of defects. Common classification networks include the Vgg series [13], Resnet series [14, 15], and ShuffleNet [16, 17] series.

ShuffleNet [16] is a lightweight model designed for high inference efficiency, incorporating two new modules, pointwise group convolution and channel shuffle, to reduce computational cost. Building on the ShuffleNetV2 [17] framework, Liang et al. [18] proposed using ShuffleNetV2 for plastic container inspection and applied it in a practical industrial inspection system. Object detection models aim to answer the questions "where is the defect and what is it?" by locating and categorizing defects. Commonly used models include the RCNN [19, 20, 21] and YOLO [22, 23, 24] series. In the literature, Tao et al. [25] adopted Faster R-CNN with a cascaded structure for power line insulator defect detection. Hu and Wang [26] developed a new PCB surface defect detection model based on Faster R-CNN, using ResNet50 as the backbone. Commonly used segmentation networks include FCN [27], Unet [28], and the DeepLab [29, 30, 31] series. Mask RCNN [21] is an advanced model that extends Faster R-CNN by integrating object detection and instance segmentation in a two-stage process. Xiao et al. [32] proposed an improved version of Mask RCNN, called IPCNN, which first uses a deep residual neural network to map the image to CNN features. These features are processed by a feature pyramid to generate pyramid features, which are then fed into an RPN to generate defect bounding boxes and classify them. Finally, FCN is used to generate a defect mask within the defect bounding box. In the field of surface defect detection of industrial products, supervised methods are currently the most widely used deep learning techniques due to their high accuracy and good adaptability. However, the drawbacks of these methods have become increasingly apparent in practical applications, particularly the substantial workload associated with labeling datasets, especially in high-precision scenarios. Additionally, the ongoing advancement of industrial standards has led to a decrease in defective samples, which presents further challenges for supervised methods.

2.2 Unsupervised Anomaly Detection Using Normal Samples

With the proposal of the industrial anomaly detection datasets, such as the MVTec AD datasets [33], the MVTec LOCO dataset [34], and the Real-ID D3 dataset [35], the research on industrial unsupervised anomaly detection has gained significant attention. The MVTec AD dataset is a widely used benchmark for evaluating anomaly detection methods, containing 15 categories with various defects. The MVTec LOCO dataset includes both structural and logical anomalies. It contains 3644 images from five different categories inspired by real-world industrial inspection scenarios. Real-ID 3D is a high-precision multimodal dataset that uniquely incorporates an additional pseudo3D modality generated through photometric stereo, alongside high-resolution RGB images and micrometer-level 3D point clouds. Unsupervised learning [36] algorithms are designed to analyze and cluster unlabeled datasets. These algorithms identify hidden structures or data clusters without human annotations by recognizing similarities and differences, making them widely applicable in computer vision tasks. However, the irregularity of defect data limits the effectiveness of standard unsupervised learning methods. Given that normal images of product surfaces are readily available, anomaly detection (AD) [37], which learns exclusively from normal images, has been proposed as a new unsupervised paradigm. The objective of AD is to develop computational models and methods that learn "what is normal" and thereby identify and locate defects. Anomaly detection has been a significant research topic for decades, with pioneering work dating back to the 1960s [38]. This method has been explored across various disciplines and applications, including image, video, text, and audio data. Image anomaly detection, an essential computer vision task, identifies abnormal regions in images and holds great potential for applications in the medical and manufacturing fields. Image anomaly detection can be performed at either the image level or the instance level.

In recent years, deep learning has demonstrated significant capabilities. However, applying deep learning algorithms using supervised learning to practical defect detection is challenging due to the scarcity of anomaly data and the highly imbalanced distribution between normal and abnormal classes. Recently, deep learning-based anomaly detection methods have been introduced, offering improved performance. Since this study primarily explores deep-learning-based anomaly detection, the term "anomaly detection" will be used for brevity in the following sections. Unsupervised anomaly detection is generally categorized into reconstruction-based and embedded feature-based methods.

2.2.1 Reconstruction-based Anomaly Detection

The first category is "image reconstruction-based methods," which represent the most fundamental approaches in anomaly detection (AD). These methods [11, 39, 40, 41, 42] are based on the principle that an encoder-decoder model is trained to reconstruct only normal images. When an abnormal image is input, the model typically fails to accurately reconstruct the anomalous regions. Therefore, the discrepancy between the input and reconstructed images indicates the presence of anomalies. **Auto-encoder (AE):** Vanilla auto-encoders (AEs) have been used for decades for feature dimensionality reduction. Masci et al. [43] first combined CNNs with AEs to create CAE, demonstrating superior performance on digit and object recognition benchmarks in 2011. With the rapid development of CNNs, improvements and applications of CAEs have increased, making it a popular framework in computer vision. One of the most prominent applications of AEs is anomaly (defect) detection. Hasan et al. [44] were the first to use CAE for video anomaly detection, pioneering the application of AEs in reconstruction tasks for unsupervised anomaly detection. Baur et al. [45] designed an unsupervised anomaly segmentation AE to detect brain anomalies in MR images. Reconstruction-based methods for industrial anomaly detection in CAE follow this AE framework. For instance, Mei et al. [46] employed CAE to reconstruct fabric images and used the residual reconstruction map for in-

dustrial fabric defect detection. Recognizing the difficulty of directly reconstructing entire fabric images with tiny defects, they focused on reconstructing image patches at various levels of the image feature pyramid. Youkachen et al. [47] used AE for anomaly detection on hot-rolled steel strip surfaces, while Chow et al. [48] presented a similar CAE application for structural concrete defect detection.

2.2.2 Representation-based Anomaly Detection

Recently, pre-trained models on natural image datasets such as ImageNet [49], as well as models trained through general computer vision tasks, have become popular feature extractors, demonstrating impressing performance. Representation-based methods for anomaly detection build an feature space for normal samples, followed by estimating features using various distribution models, such as Gaussian density estimation (GDE) [50, 51, 52] and normalizing flow (NF) [53, 54, 55, 56], as well as non-parametric models like KNN [57, 58, 59] and KDE [60, 61]. KNN-based anomaly detection collects a dictionary of normal features from a pre-trained model and then performs KNN retrieval and distance measurement. DN2 [59] was the first method to combine KNN with a pre-trained CNN feature extractor, showing promising results in image anomaly detection. Subsequently, SPADE [58] extracted patch-level features and performed patch-level anomaly detection and localization using KNN. PatchCore [57], following this approach, built a patch-level feature dictionary referred to as a feature set. Additionally, it collects multi-level features for better representation and employs a Coreset algorithm [62] to reduce Computing costs. Yao et al. [63] propose a simple but effective framework (called ResAD) that can be irectly applied to detect anomalies in new product categories. The main insight is to learn the esidual feature distribution rather than the initial feature distribution. In this way, ResAD ignificantly reduces feature variations. Wang et al. [64] proposed a istribution Prototype Diffusion Learning (DPDL) method aimed at enclosing normal samples within a compact and discriminative distribution space. Zhu et al. [65] introduced a novel approach, namely Anomaly Heterogeneity Learning

(AHL), that simulates a diverse set of heterogeneous anomaly distributions and then utilizes them to learn a unified heterogeneous abnormality model in surrogate open-set environments.

2.2.3 Synthetic Defect-based Anomaly Detection

Pre-trained CNNs are not ideal for anomaly detection due to the domain bias between pre-trained tasks and current task. To mitigate this problem, many researchers have proposed using self-supervised learning (SSL) [66] to create proxy tasks that reduce domain bias. SSL methods learn features from unlabeled images. Among these, synthetic defect-based methods [67, 68, 69, 70, 71], which generate synthetic defect samples through image editing, are some of the most commonly used proxy tasks for anomaly detection. Image editing is a widely adopted data augmentation technique in supervised learning [72, 73]. A straightforward technique [69, 70] involves randomly removing regions of various sizes and filling them with customized values. However, these methods often produce artificial defects that fail to accurately mimic real defects, resulting in biased models with poor generalization. Improved techniques fall into two categories. The first focuses on increasing the complexity of the training task [71, 67, 74, 75]. For example, Zavrtanik et al. [74] and Yan et al. [75] increased training difficulty by removing significant portions of the image during training, reducing information redundancy and making the reconstruction task more challenging. This approach led to better anomaly detection performance. Additionally, Tan et al. [71] first proposed using image patches from different images within the same dataset as defects. Similarly, Li et al. [67] introduced an SSL method called Cutpaste, where a region is cropped from a defect-free image and pasted onto another to create a synthetic defect. These methods treat normal patches from other images as defects, thereby increasing the complexity of the learning task and improving feature representation. The second category aims to generate more diverse defects. Theoretically, the more diverse the synthetic defects, the smaller the domain bias. As a result, some methods incorporate different

defect distributions, sizes, brightness levels, and shapes to create a wider variety of synthetic defects [68, 76]. For instance, Zavrtanik et al. [68] generated more realistic defects by cropping regions from various texture images in public datasets and using them as defective fills. Schlüter et al. [76] proposed a Poisson fusion method to address the visible discontinuity at defect margins caused by pasting one part of an image onto another.

2.3 Few-shot Defect Multi-Classification

2.3.1 Few-shot Learning

Given the challenge of acquiring a substantial number of labeled samples in many practical scenarios, the applicability of deep neural network models is often significantly constrained. Few-shot learning addresses this by focusing on the critical challenge of training models to achieve high performance with only a limited set of labeled samples, a necessity for advancing machine learning in various domains. In few-shot learning, each category is represented by a small set of training samples, referred to as the support set, with the objective of classifying the query set based on the model's training on this support set. However, the scarcity of training data increases the risk of overfitting, which can degrade model performance. To mitigate this, few-shot learning typically involves an additional training set, known as the base classes, which contains categories distinct from those in the support set. The model is initially trained on these base classes, and when presented with a new task involving novel classes, it quickly adapts using the support set, thereby achieving good performance on the new task. This approach is analogous to the human cognitive process, where individuals can swiftly learn to identify a new object from a few examples after building up prior knowledge.

Few-shot learning has been applied in various fields, including medical image recognition, defect detection in products, robotic action learning, speech synthesis, and more. These models have demonstrated impressive performance across different

datasets, highlighting the potential of few-shot learning in defect classification tasks. However, traditional methods often involve training a base model on the base classes and then fine-tuning it on novel classes, which limits their ability to handle all classes simultaneously. The proposed MVREC model, utilizing the CLIP model, addresses this limitation by allowing evaluation on all classes within any dataset, making it more reflective of real-world scenarios.

2.3.2 CLIP-based Few-shot Classification

The most common few-shot methods include meta-learning and metric learning. Meta-learning methods learn a model that can quickly adapt to new tasks with minimal training data. Metric learning methods learn a distance metric that can effectively measure the similarity between samples. Recently, large language models and multi-modal pre-training models, such as GPT [77, 78] and CLIP [79], have emerged as powerful tools. Related research [80, 81] has been applied to defect detection tasks, showing impressive performance. Numerous adapter-based methods have been proposed to adapt the CLIP model to specific tasks with few samples, such as CLIP-Adapter [82], Tip-Adapter [83], CoOp [84], and SuS-X [85]. To enable CLIP to focus on specific regions within the entire image, various methods [86, 87, 88, 89] have been explored. AlphaCLIP is an innovative enhancement of the CLIP model, designed to improve its ability to focus attention on specific regions [89]. This architecture enables AlphaCLIP to provide precise control over the emphasis of image content. In this work, AlphaCLIP is used to generate the region-context feature for the defect instance.

2.4 Continual Learning-based Fabric Inspection Model

2.4.1 Fabric Inspection Models using Deep Learning

The current techniques for fabric defect detection include structural, spectral, statistical, model-based, and learning-based approaches [90]. Structural methods view fabric textures as assemblies of textural primitives, typically identifying defects through a two-step process: first, by detecting basic fabric textures, and second, by learning the overall texture patterns. However, these methods often struggle with irregular fabric textures. Spectral methods, on the other hand, identify defective regions by converting the original image into the spectral domain and analyzing the filter response energy. Techniques such as Fourier [91], Gabor [92], and wavelet transforms [93] are included under spectral methods, effectively leveraging global information from fabric images. Model-based methods address defect detection through modeling and decomposition techniques. Although spectral and statistical methods face challenges in inspecting stochastic surface variations in randomly textured fabrics, model-based methods, such as auto-regressive models and Gauss Markov random fields (MRF), can offer improved performance in these scenarios. Recent advancements in deep learning have become crucial for developing algorithms that address image processing challenges in quality inspection within the textile and apparel industry. For instance, Jing et al. [94] applied the deep convolutional neural network YOLOv3, originally successful in general object detection tasks, to fabric defect detection. Additionally, Zhu et al. [95] modified the DenseNet architecture and developed a tailored cross-entropy loss function aimed specifically at defect detection within edge computing environments. This study presents an innovative prototypical network crafted to improve fabric defect classification, particularly in cases where class distributions are imbalanced. The proposed approach, evaluated against five established models using a commercial fabric defect dataset, achieved a leading classification accuracy of 96.04% across seven defect categories.

2.4.2 Continual Learning with Pre-Trained Models

With the swift progress of deep neural networks, deep learning models have exhibited exceptional potential across numerous applications. However, in practical situations, data frequently arrives in a streaming manner, necessitating the development of learning systems that can adapt and evolve continuously. The emerging field of pre-training techniques has recently offered promising opportunities in this area. Leveraging pre-trained models (PTMs) developed from extensive datasets and advanced techniques [96] has proven highly effective for continual learning (CL). These PTMs exhibit robust generalization across a wide array of downstream tasks, leading to the growing popularity of PTM-based CL in research. The integration of PTMs into CL generally follows three main approaches: prompt-based methods, representation-based methods, and model mixture-based methods. In prompt-based methods [97], visual prompts are fine-tuned while keeping the pre-trained weights unchanged. Conversely, model mixture-based methods generate multiple models throughout the learning process and apply strategies like model merging, ensembling, and other combination techniques to derive the final prediction. Representation-based methods operate under the assumption that the knowledge required for new tasks is already encapsulated within the PTMs, exploiting their generalization capabilities to construct downstream models directly. This section is dedicated to exploring representation-based methods in detail.

Inspired by representation learning, SimpleCIL [98] introduces a straightforward approach to handle continuous data streams by freezing pre-trained weights and computing a central representation (prototype) for each class. This method captures the most representative pattern by averaging the embeddings of samples within the same class, which is then used as a classification template during inference. SimpleCIL replaces the classifier weight for each class with its corresponding prototype and employs a cosine classifier for prediction. Despite its simplicity, this approach yields impressive results, demonstrating that pre-trained models (PTMs) possess sufficiently generalizable representations that can be directly applied to downstream

tasks. Similar results have been observed in other studies [99], with [100] extending this concept to large language models. Building on the strong generalization capabilities of PTMs, ADAM [98] evaluates performance on new classes by comparing prototype-based classifiers to fully fine-tuned models. The findings suggest that while PTMs are highly generalizable, they may lack the specific information needed for optimal performance on new downstream tasks. Consequently, ADAM advocates fine-tuning PTMs using parameter-efficient modules, such as prompts [101] or adapters [102], and combining features from both the pre-trained and adapted models. This strategy effectively bridges the domain gap between pre-trained and downstream datasets, leading to improved performance over SimpleCIL. More recently, EASE [103] further enhances performance by integrating feature representations from multiple task-specific backbones, achieving state-of-the-art results. It introduces a semantic mapping strategy to complement the classifier, addressing challenges posed by the expanding feature space and previous classifiers. Building on ADAM, RanPAC [104] discovers that prototypes often exhibit class-wise correlations and suggests using an online LDA model to remove the intra-class correlations, thereby improving class separability. Additionally, RanPAC introduces a random projection layer to align the feature distribution with a Gaussian fit, projecting features into a higher-dimensional space. Furthermore, LayUP [105] identifies that the strong representational capability extends into the deeper layers of transformer blocks. Representation-based methods are designed to fully leverage the features extracted by pre-trained models, which have demonstrated strong performance across various tasks. This approach offers several key advantages: it provides intuitive and interpretable recognition models by capturing the most typical pattern of each class, facilitates the assessment of baseline performance in PTM-based continual learning, and supports a lightweight update process by freezing the backbone and only updating classifier weights, making it practical for real-world applications.

2.5 Chapter Summary

This chapter reviews the use of deep learning in industrial product surface inspection from a learning perspective. Then methods for unsupervised anomaly detection and few-shot defect multi-classification are introduced. Furthermore, the continual learning-based methods and fabric inspection models are discussed. However, the existing anomaly detection methods do not fully consider the domain bias between pre-trained models and industrial anomaly detection tasks and the intra-image distribution bias in industrial images, which limits their performance. The existing defect Few-shot defect multi-classification methods is rarely and do not exploit the pre-trained CLIP model and the region context feature for defect instance. The current fabric inspection models ,mainly focuses on the deep learning-based defect detection methods, and the continual learning-based fabric inspection model is not fully explored. The next chapter will introduce the methodology of this work to address these limitations.

Chapter 3

Methodology

This chapter presents the proposed industrial product surface inspection methodology and introduces the related deep learning methods. Additionally, the datasets and the metrics used are presented.

3.1 Overview of the proposed Computer Vision Model for Industrial Product Surface Inspection

The goal of this study is to develop a Computer Vision Model for Industrial Product Surface Inspection, with a focus on fabrics in the textile and apparel industry. To achieve this goal, four specific tasks are designed. The overall research process, consisting of four phases, is illustrated in Figure 3.1. The framework includes four phases: 1) Dataset Construction, 2) Unsupervised Anomaly Detection, 3) Few-shot Learning-based Defect Classification, and 4) A Practical End-to-End Fabric Detection Model Using Continual Learning.

Phase 1. Dataset Construction:

To evaluate the performance of the developed algorithms, both public and self-collected datasets are utilized.

Phase 2. Industrial Anomaly Detection:

This phase involves learning the distribution of normal samples using pre-trained models and detecting anomalies in industrial product images, which is framed as a one-class classification problem. This task addresses domain bias between the pre-trained model and industrial product images, models the distribution of normal samples, and measures the distance between normality and test samples.

Phase 3. Few-shot Defect Multi-classification:

This phase focuses on solving the defect classification problem at the instance level after learning from few-shot samples. Several techniques are employed, including multi-view augmentation, region context representation, fine-tuning, and adapter-based models. The model is trained with a support set and evaluated using a query set and few-shot samples.

Phase 4. Continual Learning-based Fabric Inspection Model:

This phase addresses the challenges of pattern shift and inspection criteria shift in fabric inspection tasks. In practical scenarios, the model may encounter unseen fabric patterns and defects. To evaluate the model, both a self-collected solid color fabric dataset and a novel fabric dataset are used. The model is first pre-trained on the solid color fabric dataset and then tested on the novel fabric dataset using available samples. By adapting the model with few-shot samples, it can effectively detect unseen defects in the novel fabric dataset.

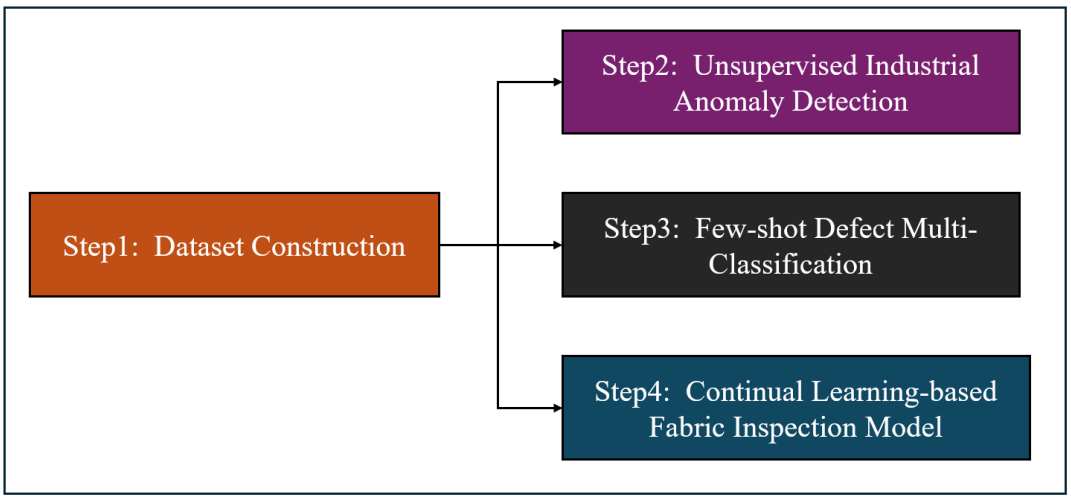

Figure 3.1: Architecture of the proposed intelligent framework.

3.2 Dataset Construction

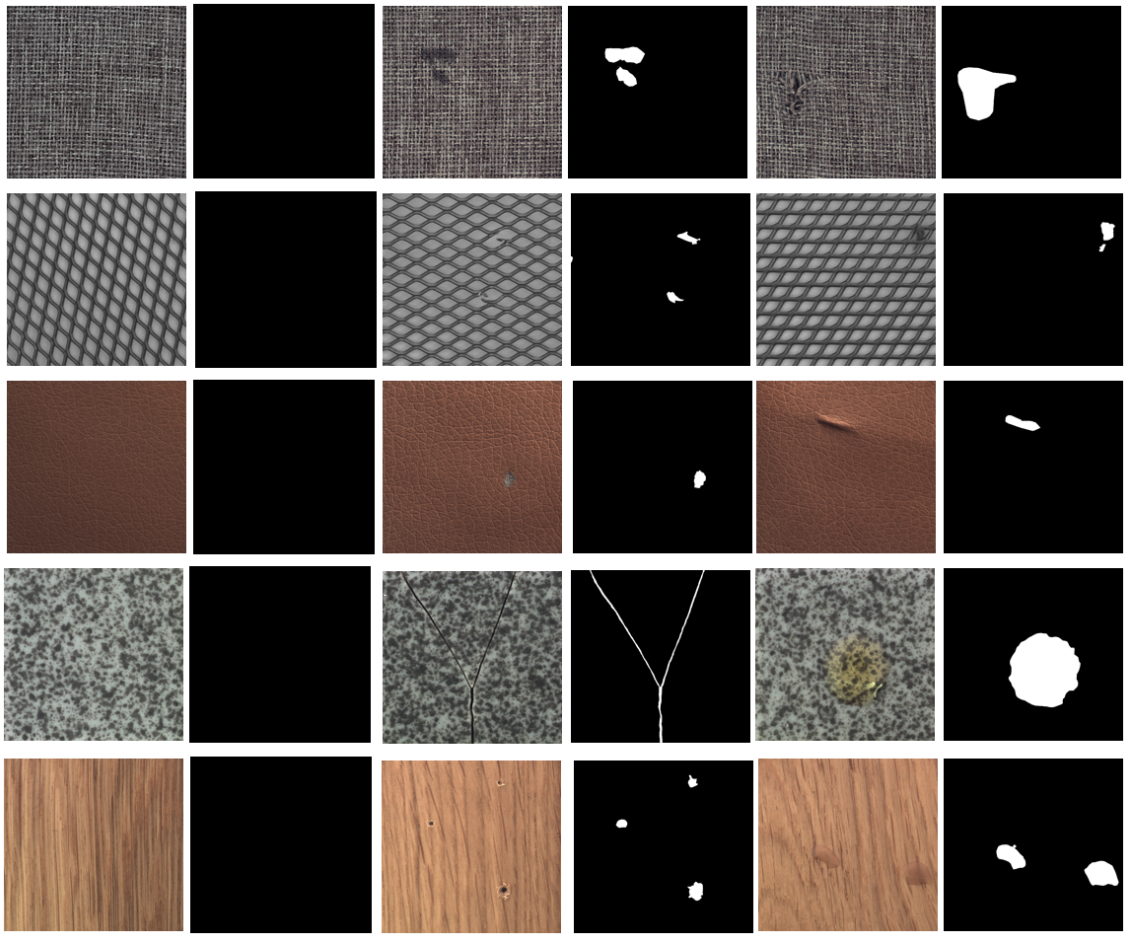

Figure 3.2: Visualization of the MVTec AD Dataset. The dataset consists of 15 different product types.

In this section, datasets for training and evaluating the performance of related methods and the algorithms of the proposed intelligent model are described. Some public industrial product defect datasets and self-collected fabric datasets are introduced in this section.

Public datasets

The public datasets include the MVTec AD dataset [33], MVTec LOCO dataset [106], and BTAD dataset [107] are used to evaluate the performance of the developed anomaly detection model. These datasets are the most widely used benchmarks for industrial anomaly detection. MVTed-FS based on MVTed AD is built to evaluate the performance of the developed few-shot classification model,

with more details shown in chapter 5. Additionally, four other datasets with classifcation annotations also are used to evaluate the performance of the developed defect classification model, including NEU-DET [108], PCB Defect Dataset [109], Magnetic Tile Surface Defects (MTD) [110], AITEX Fabric defect [111].

MTFabric dataset

Owing to the lack of fabric defect datasets focusing our the problems, this study constructs one Mulit-type Fabric (MTFabric) dataset, which contains three types of fabric with different patterns and inspection standards. We categorize these as Fabric Type A, Fabric Type B, and Fabric Type C. Fabric Type A represents a knitted fabric with a consistent pattern (see Table 3.1). Fabric Type B, a printed fabric (see Table 3.2), and Fabric Type C, characterized by irregular stripes (see Table 3.3), are sourced from the publicly available Guangdong dataset [112]. This dataset is designed for the evaluation of intelligent fabric inspection models, with instance-level bounding boxes applied to label defects. Each type is divided into two primary subsets: the support set and the query set. The support set comprises a mixture of normal images and a limited number of defective images, while the query set includes the remaining images. The model is initially trained on the support set and subsequently tested on the query set, with performance assessed using the few-shot defective samples.

Figure 3.3: Visualization of the MTFabric Dataset. The dataset consists of three fabric types with differnt patterns and inspection standards.

Split	Image			Defect						
	Total	Normal	Defective	StopMark	OilNeedle	OilYarn	HorizontalStrip	Knot	Stain	StainPoint
support set	218	142	76	33	11	11	18	4	6	6
query set	120	52	68	18	5	11	25	6	6	7

Table 3.1: Details of fabric type A in novel fabric dataset.

Split	Image			Defect Count Map			
	Total	Normal	Defective	Seam	Seam Impression	Crease	Pattern Hair
query set	141	47	94	61	23	13	3
support set	187	71	116	83	28	16	5

Table 3.2: Details of fabric type B in novel fabric dataset.

Split	Image			Defect Count Map			
	Total	Normal	Defective	Seam	Stain	Crease	Pattern Hair
support set	206	45	161	90	83	41	11
query set	258	127	131	68	69	27	11

Table 3.3: Details of fabric type C in novel fabric dataset.

3.3 Industrial Anomaly Detection

In industrial defect detection, the combination of pre-trained models, fine-tuning techniques, and distance-based anomaly detection methods is pivotal for achieving high accuracy and efficiency. Leveraging pre-trained models on large datasets like ImageNet [49], followed by domain-specific fine-tuning, enables models to adapt to the unique characteristics of industrial images. Additionally, distance measurement techniques, such as k-Nearest Neighbors (k-NN) with Euclidean distance, provide robust mechanisms for identifying anomalies, ensuring reliable and precise defect detection in industrial settings.

3.3.1 Pre-trained Models and Fine-tuning

Pre-trained models have become a cornerstone in the field of computer vision (CV), significantly enhancing the efficiency and performance of downstream tasks. These models, typically trained on large and diverse datasets such as ImageNet, capture a wide range of features, from low-level edges to high-level object representations. ImageNet, in particular, is a widely recognized benchmark in the CV community, consisting of millions of images across thousands of categories. By pre-training on such a comprehensive dataset, models develop a robust ability to generalize across various visual tasks.

When applied to downstream tasks, pre-trained models offer several advantages. First, they reduce the need for extensive labeled data, as the pre-trained weights provide a strong starting point for further fine-tuning. This is particularly beneficial for specialized tasks where labeled data may be scarce or expensive to obtain. Second, using pre-trained models accelerates the training process since the model already possesses a foundational understanding of visual features. Fine-tuning these models for specific tasks, such as defect detection in industrial products, allows the network to adapt its learned features to the nuances of the new task, often leading to improved accuracy and efficiency compared to training from scratch.

However, while pre-trained models provide a solid foundation, they often perform suboptimally when directly applied to downstream tasks due to domain bias. Domain bias occurs when the data distribution of the pre-trained model's source domain (e.g., ImageNet) differs significantly from that of the target domain (e.g., industrial product images). To mitigate this issue, fine-tuning is employed, which involves retraining the pre-trained model on a smaller dataset specific to the target domain. This process can be mathematically expressed as an optimization problem:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}_i; \theta), y_i) \tag{3.1}$$

where θ represents the parameters of the pre-trained model, $\mathbf{x}_i$ and y_i are the input images and their corresponding labels in the target domain, and $\mathcal{L}$ is the loss function used to measure the difference between the predicted and actual labels. The goal is to adjust the pre-trained weights θ to minimize the loss on the target domain data.

In the context of defect detection, fine-tuning is particularly important because the nature of defects in industrial products can vary greatly from the objects typically found in general-purpose datasets like ImageNet. To further reduce domain bias, a common approach is to fine-tune the model using a proxy task that involves classifying between normal images and synthetically generated defect images. This approach allows the model to adapt its learned features more effectively to the spe-

cific characteristics of the target domain.

The training process for this proxy task typically employs a cross-entropy loss function, which is widely used for classification tasks. The cross-entropy loss for this task can be defined as:

$$\mathcal{L}_{\text{cross-entropy}} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)] \tag{3.2}$$

where y_i is the true label of the i-th sample (with 1 for defect and 0 for normal), $\hat{y}_i$ is the predicted probability of the i-th sample being a defect, and N is the total number of samples. This loss function measures how well the model's predictions match the true labels and is minimized during fine-tuning to improve the model's classification accuracy.

By treating the differentiation between normal and defect samples as a proxy task and using synthetic defects, the model can better learn to distinguish between defect-free and defective regions, thereby enhancing its ability to detect real-world anomalies.

This strategy not only adapts the model to the specific characteristics of industrial images but also leverages the robustness of pre-trained features to improve defect detection accuracy. Consequently, the adoption of pre-trained models, combined with fine-tuning on such proxy tasks, has become a standard practice in computer vision, driving advancements across a wide array of applications.

3.3.2 Normality Representation

Normality is represented by features extracted from defect-free samples, which serve as the baseline for identifying deviations in defect detection tasks. These features are typically derived from a fine-tuned model trained on images within the target domain that contain no defects. By encoding these normal features, the model develops a comprehensive understanding of what constitutes 'normal' in the context of industrial product images.

The extracted features are stored in a memory bank, denoted as $\mathcal{M}$, which serves

as a reference repository:

$$\mathcal{M} = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_N\} \tag{3.3}$$

where $\mathbf{f}_i$ represents the feature vector of the i-th normal sample, and N is the total number of stored normal samples.

During the defect detection process, the features of new samples $\mathbf{f}_{\text{new}}$ are compared against the memory bank $\mathcal{M}$. If these features significantly deviate from those in $\mathcal{M}$, the sample is flagged as potentially defective. This method ensures consistent and accurate identification of anomalies by leveraging the fine-tuned feature space.

3.3.3 Distance Measurement

In the context of anomaly detection, distance measurement is critical for assessing how much a test sample's features deviate from those of the normal samples stored in the memory bank. One of the most common approaches is to use the k-Nearest Neighbors (k-NN) algorithm in conjunction with Euclidean distance to quantify this deviation.

The k-NN algorithm identifies the k closest normal samples to the test sample within the feature space. The proximity of these samples is determined using the Euclidean distance metric. The Euclidean distance $d(x, y)$ between two feature vectors $\mathbf{x}$ and $\mathbf{y}$ is calculated as:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{3.4}$$

where x_i and y_i represent the i-th features of the vectors $\mathbf{x}$ and $\mathbf{y}$, respectively, and n is the number of features.

Once the k nearest neighbors are identified, the anomaly score for the test sample is computed based on the distances to these neighbors. A common approach is to use the average distance to the k nearest neighbors:

$$\text{Anomaly Score} = \frac{1}{k}\sum_{j=1}^{k} d(\mathbf{x}, \mathbf{y_j}) \tag{3.5}$$

where $\mathbf{y_j}$ represents the j-th nearest neighbor to the test sample $\mathbf{x}$. A higher anomaly score indicates a greater deviation from the normal samples, suggesting that the test sample is more likely to be an anomaly.

The effectiveness of k-NN with Euclidean distance in anomaly detection stems from its simplicity and interpretability. By focusing on the direct, straight-line distance between feature vectors, this method provides an intuitive measure of similarity. Moreover, by averaging the distances to the k nearest neighbors, the model can smooth out noise and achieve a more robust assessment of whether a sample is anomalous.

3.4 Few-shot Defect Mulit-Classifcation

3.4.1 Few-Shot Classification

Few-shot classification aims to classify objects using only a small number of labeled examples. In the conventional N-way K-shot setting, models typically categorize classes into base classes and novel classes. Base classes are those with ample labeled data available during training, enabling the model to learn robust feature representations. Novel classes, however, are introduced during testing with only K labeled examples per class, challenging the model to generalize from minimal data.

Formally, let $\mathcal{C}_{\text{base}}$ and $\mathcal{C}_{\text{novel}}$ denote the sets of base and novel classes, respectively, where $\mathcal{C}_{\text{base}} \cap \mathcal{C}_{\text{novel}} = \emptyset$. The model is trained on tasks $\mathcal{T}_{\text{train}}$, each involving a support set $\mathcal{S}_i$ and a query set $\mathcal{Q}_i$, drawn from $\mathcal{C}_{\text{base}}$. During testing, the model is evaluated on tasks drawn from $\mathcal{C}_{\text{novel}}$, where it must classify novel classes with limited examples.

In our approach, however, this study removes the distinction between base and novel classes. Instead, this study evaluates the model across all classes, relying solely on the few examples provided per class, regardless of whether the class was seen

during training. This unified approach leverages the model's ability to generalize across all classes:

$$\mathcal{S}_i = \{(x_j, y_j)\}_{j=1}^{N \times K}, \quad y_j \in \mathcal{C}_{\text{all}} \tag{3.6}$$

$$\mathcal{Q}_i = \{(x_l, y_l)\}_{l=1}^{L}, \quad y_l \in \mathcal{C}_{\text{all}} \tag{3.7}$$

where $\mathcal{C}_{\text{all}}$ represents the complete set of classes without distinguishing between base and novel classes. This strategy allows the model to be evaluated across all available classes, relying purely on the small number of examples per class, thereby streamlining the classification process and potentially enhancing its generalization capabilities across a broader range of classes.

3.4.2 Contrastive Language-Image Pretraining Model

Contrastive Language-Image Pretraining (CLIP) [79] is a model that learns a joint embedding space for both images and text. By aligning image and text representations in the same space, CLIP enables a range of tasks such as zero-shot and few-shot learning without requiring task-specific fine-tuning. CLIP leverages large-scale datasets consisting of image-text pairs to train its model, resulting in a versatile and generalizable representation.

The CLIP model consists of two main components: an image encoder $f_{\text{img}}(\cdot)$ and a text encoder $f_{\text{text}}(\cdot)$, which project images and texts into a shared embedding space. During training, CLIP optimizes a contrastive loss function that encourages image-text pairs to have similar embeddings while pushing apart embeddings of unrelated pairs.

Zero-Shot Learning with CLIP

CLIP's zero-shot learning capability leverages the model's ability to understand and align image and text representations without the need for additional training on the specific downstream task. In a zero-shot setting, the model is provided with a set of class labels described in natural language and is tasked with classifying images

into these classes without having seen any labeled examples for these classes during training.

Given an image x_{query}, the zero-shot classification is performed as follows:

1. Encode the image using the image encoder:

$$\mathbf{z}_{\text{img}} = f_{\text{img}}(x_{\text{query}}) \tag{3.8}$$

2. Encode each class label using the text encoder. For each class c_j with description t_j, compute:

$$\mathbf{z}_{\text{text}}^{j} = f_{\text{text}}(t_j) \tag{3.9}$$

3. Compute the similarity between the image embedding and each class label embedding:

$$\text{sim}_j = \text{sim}(\mathbf{z}_{\text{img}}, \mathbf{z}_{\text{text}}^{j}) \tag{3.10}$$

4. Assign the image to the class with the highest similarity score:

$$y_{\text{pred}} = \arg\max_{j} \text{sim}_j \tag{3.11}$$

Few-Shot Learning with CLIP

For few-shot learning, CLIP can be adapted by utilizing a small number of labeled examples to refine the alignment between the image and text embeddings. While CLIP can be used directly in a zero-shot manner, leveraging a few labeled examples can help improve its performance on the target task.

The few-shot learning process with CLIP typically involves the following steps:

1. **Support Set Construction**: Collect a small support set $\mathcal{S}$ consisting of K labeled examples for each of the N classes. Each support example includes an image x_j and its corresponding class label y_j described by text t_j.

2. **Feature Extraction**: Encode the images in the support set and their corre-

sponding text descriptions using the CLIP encoders:

$$\mathbf{z}_{\text{img}}^{j} = f_{\text{img}}(x_j), \quad \mathbf{z}_{\text{text}}^{j} = f_{\text{text}}(t_j) \tag{3.12}$$

3. **Prototypical Representations**: Compute a prototypical representation for each class by averaging the embeddings of the support examples for that class:

$$\mathbf{z}_{\text{proto}}^{c} = \frac{1}{K} \sum_{j=1}^{K} \mathbf{z}_{\text{img}}^{j} \tag{3.13}$$

4. **Query Classification**: For a given query image x_{query}, encode it using the image encoder:

$$\mathbf{z}_{\text{query}} = f_{\text{img}}(x_{\text{query}}) \tag{3.14}$$

Then, compute the similarity between the query image embedding and each class prototype:

$$\text{sim}_c = \text{sim}(\mathbf{z}_{\text{query}}, \mathbf{z}_{\text{proto}}^{c}) \tag{3.15}$$

Finally, assign the query image to the class with the highest similarity score:

$$y_{\text{pred}} = \arg\max_{c} \text{sim}_c \tag{3.16}$$

This few-shot classification strategy allows CLIP to adapt to new tasks with limited labeled data, leveraging its pretrained joint embedding space to make accurate predictions.

3.4.3 Region Context-based Models

Unlike dense prediction tasks such as object detection and instance segmentation, which can predict the positions and categories of multiple instances within an image, classification networks typically provide an image-level prediction. This approach does not account for the presence of multiple instances within a single image, potentially leading to a loss of information regarding individual instances. To address this

limitation, region context-based models have been proposed. These models incorporate positional information as a prompt to predict the target's information while preserving the contextual information around the target.

$$\mathbf{P}(I) = \mathbf{f}(R, C) \tag{3.17}$$

In this equation, $\mathbf{P}(I)$ represents the predicted information for the image I, $\mathbf{f}$ denotes the function of the model, R indicates the region of interest within the image, and C represents the contextual information surrounding that region.

For example, the Segment Anything Model (SAM) network uses prompts in the form of points, bounding boxes, or masks to guide the segmentation process. By incorporating these prompts, the network is able to preserve the contextual information around the target, leading to more accurate predictions for specific regions within the image.

$$\mathbf{T}(R, P) = \sum_{i=1}^{n} \mathbf{w}_i \cdot \mathbf{g}(R_i, P_i) \tag{3.18}$$

Here, $\mathbf{T}(R, P)$ denotes the target information predicted by the model for region R with prompt P, where R_i and P_i are the respective regions and prompts at position i, $\mathbf{w}_i$ represents the weight assigned to the contextual information, and $\mathbf{g}$ is a function that combines the region and prompt information.

These models effectively bridge the gap between image-level predictions and instance-level predictions, allowing for more precise and context-aware classification of multiple instances within a single image. The incorporation of region-based prompts ensures that the model can focus on specific areas of interest, enhancing its ability to discern and categorize multiple targets within the image.

3.5 Continual Learning-based Fabric Inspection Model

The Continual Learning-based fabric inspection model (CLFIM) integrates the approach outlined in the previous two phases, utilizing normal samples along with a small number of defective samples. The main differences between this chapter and the previous ones are as follows: First, this model is specifically designed for fabric inspection, with pre-training conducted on a solid-color fabric dataset. Second, it is an end-to-end model that directly performs detection and classification tasks, making it more suitable for real-world applications. This study discusses the application of end-to-end YOLOv8 for object detection, deep feature extraction for enhanced defect representation, and the use of pre-trained models (PTMs) in continual learning to ensure robust.

3.5.1 YOLOv8 for Object Detection

The first phase involves the application of YOLOv8 (You Only Look Once, Version 8), a SOTA one-stage object detection model known for its efficiency. YOLOv8 takes an input image $\mathbf{I}$ and outputs a set of bounding boxes along with their corresponding class labels and confidence scores. Formally, the output can be represented as:

$$\text{YOLOv8}(\mathbf{I}) = \{(\mathbf{b}_i, c_i, s_i)\}_{i=1}^{n} \tag{3.19}$$

where $\mathbf{b}_i = (x_i, y_i, w_i, h_i)$ denotes the bounding box for the i-th detected defect, with x_i and y_i being the coordinates of the center of the bounding box, and w_i and h_i representing the width and height, respectively. c_i is the class label assigned to the i-th defect, and s_i is the confidence score indicating the probability that the detected defect belongs to class c_i.

The output from YOLOv8 provides the preliminary identification of potential defects within the image, with each detected defect characterized by its location and category. This allows for further, more detailed analysis to confirm the presence

and nature of any defects identified.

3.5.2 Continual Learning with Pre-Trained Models

Continual learning (CL) focuses on learning from a continuous stream of data, instead of learning static dataset. Formally, given a sequence of tasks $\mathcal{T} = (T_1, T_2, \ldots)$, where each task T_i comprises a training dataset

$$D_i = \{(x_1^{(i)}, y_1^{(i)}), (x_2^{(i)}, y_2^{(i)}), \ldots\}, \tag{3.20}$$

an offline continual learning algorithm $\mathcal{A}_{CL}$ updates the weights θ using the current task D_k, such that $\theta_k = \mathcal{A}_{CL}(\theta_{k-1}, D_k)$, beginning from an initial parameter θ_0.

To mitigate the issue of forgetting in CL, many setups employ an episodic memory M_k, a limited-size subset of training data from prior tasks, such that $(\theta_k, M_k) = \mathcal{A}_{CL}(\theta_{k-1}, M_{k-1}, D_k)$. The goal is to minimize the error of θ_k across all encountered tasks $\{T_i\}_{i=1}^{k}$.

In contrast to offline CL, where all training samples in D_k are available simultaneously, online CL processes data as a stream of samples $(x_1^{(k)}, y_1^{(k)}), (x_2^{(k)}, y_2^{(k)}), \ldots$. An online CL algorithm $\mathcal{A}_{OCL}$ is thus defined as:

$$(\theta_{k,t}, M_{k,t}) = \mathcal{A}_{OCL}\left(\theta_{k,t-1}, M_{k,t-1}, (x_t^{(k)}, y_t^{(k)})\right) \tag{3.21}$$

with the same objective of minimizing the error of $\theta_{k,t}$ on $\{T_i\}_{i=1}^{k}$. As the algorithm does not retain previous samples $\{(x_s^{(k)}, y_s^{(k)}), s < t\}$ at time t, multi-epoch training on D_k is not feasible in the online CL setting.

Representation-based methods are specifically designed to maximize the utility of features extracted by pre-trained models, which have shown high effectiveness across various tasks. This approach provides several significant benefits. Firstly, class prototypes encapsulate the most representative patterns of their respective classes, making them a natural and interpretable basis for constructing recognition

models. Prototype-based classifiers also offer a simple yet effective benchmark for evaluating the performance of PTM-based continual learning (CL). Furthermore, this method usually involves freezing the backbone and only updating the classifier weights, leading to a lightweight and practical update process suitable for real-world scenarios.

3.6 Chapter Summary

This chapter has outlined the development of a computer vision model for industrial product surface inspection using unsupervised learning and few-shot learning techniques. Initially, this study introduces the public datasets and the self-collected fabric datasets utilized in this study. Following this, this study introduces the research methodology for industrial anomaly detection, which includes model pre-training, fine-tuning, normality representation, and distance measurement. Subsequently, this study discusses the research methodology for region-context-based few-shot defect multi-classification, incorporating deep feature extraction techniques and advanced algorithms such as CLIP and AlphaCIP. This methodology covers few-shot classification as well as the evaluation strategy for the proposed few-shot classification method. Finally, this study presents the research methodology for the continual learning-based fabric inspection model, which integrates the popular one-stage detection model YOLOv8 and continual learning with pre-trained models. This chapter provides a comprehensive overview of the research methodologies employed in this study, setting the stage for the subsequent chapters to explore the technical details and experimental results.

Chapter 4

Reducing Biases in Representation for Industrial Anomaly Detection

This chapter introduces an advanced methodology for industrial anomaly detection, addressing the shortcomings of current representation-based techniques. Typically, these methods follow a two-stage process: first, extracting feature representations using a pre-trained model, and second, applying distance measures to identify anomalies. Among these approaches, K-nearest neighbor (KNN) retrieval methods have shown significant potential. However, their effectiveness is often constrained by insufficient feature utilization, a consequence of domain bias in pre-trained models and variations in local density within the feature space.

To address these limitations, this chapter presents a novel method called Reducing Biases (REB) in representation. REB mitigates domain bias through the integration of a self-supervised learning task specifically tailored for enhanced domain adaptation, coupled with a defect generation strategy named *DefectMaker*, which ensures substantial diversity in synthetic defects. Furthermore, a local-density KNN (LDKNN) technique is proposed to reduce local density bias in the feature space, thereby improving detection results.

The effectiveness of the proposed method is demonstrated by achieving remarkable results, such as a 99.5% Im.AUROC on the well-known MVTec AD dataset us-

ing smaller backbone networks like Vgg11 and Resnet18. Additionally, the method achieves 88.8% Im.AUROC on the MVTec LOCO AD dataset and 96.0% on the BTAD dataset, surpassing other representation-based approaches. These results validate the method's effectiveness and efficiency, making it a viable solution for industrial applications.

4.1 Introduction

Inspired by human cognition, automatic anomaly detection [113] seeks to identify unusual patterns or disrupted structures in data predominantly trained on normal instances. This approach has found extensive application across various domains, including industrial product surface inspection [114], video analysis [115], and medical diagnosis [116, 117].

This research specifically focuses on the challenge of industrial anomaly detection, which involves identifying anomalies within industrial images. Figure 4.1 presents illustrative examples from the MVTec AD [33] and MVTec LOCO AD [106] datasets, along with corresponding detection outcomes. Otaining a substantial amount of anomalous data for training is difficult in practical scenarios. To address this problem, current studies have employed unsupervised anomaly detection techniques, utilizing auto-encoding models, Generative Adversarial Networks (GANs), and representation-based methods.

Representation-based methods for anomaly detection typically consist of two main components: a feature extractor and a distribution estimator. The feature extractor derives features from normal images or image patches using pre-trained Deep Learning (DL) models. The distribution estimator then models the normal feature representation and calculates the distance between the extracted features of a given image and the model's normal feature distribution, which serves as the anomaly score. These methods have become increasingly popular due to their simplicity and effectiveness, leveraging pre-trained models that benefit from the rapid advancements in computer vision. Many studies [118, 57, 50, 51, 58] have utilized

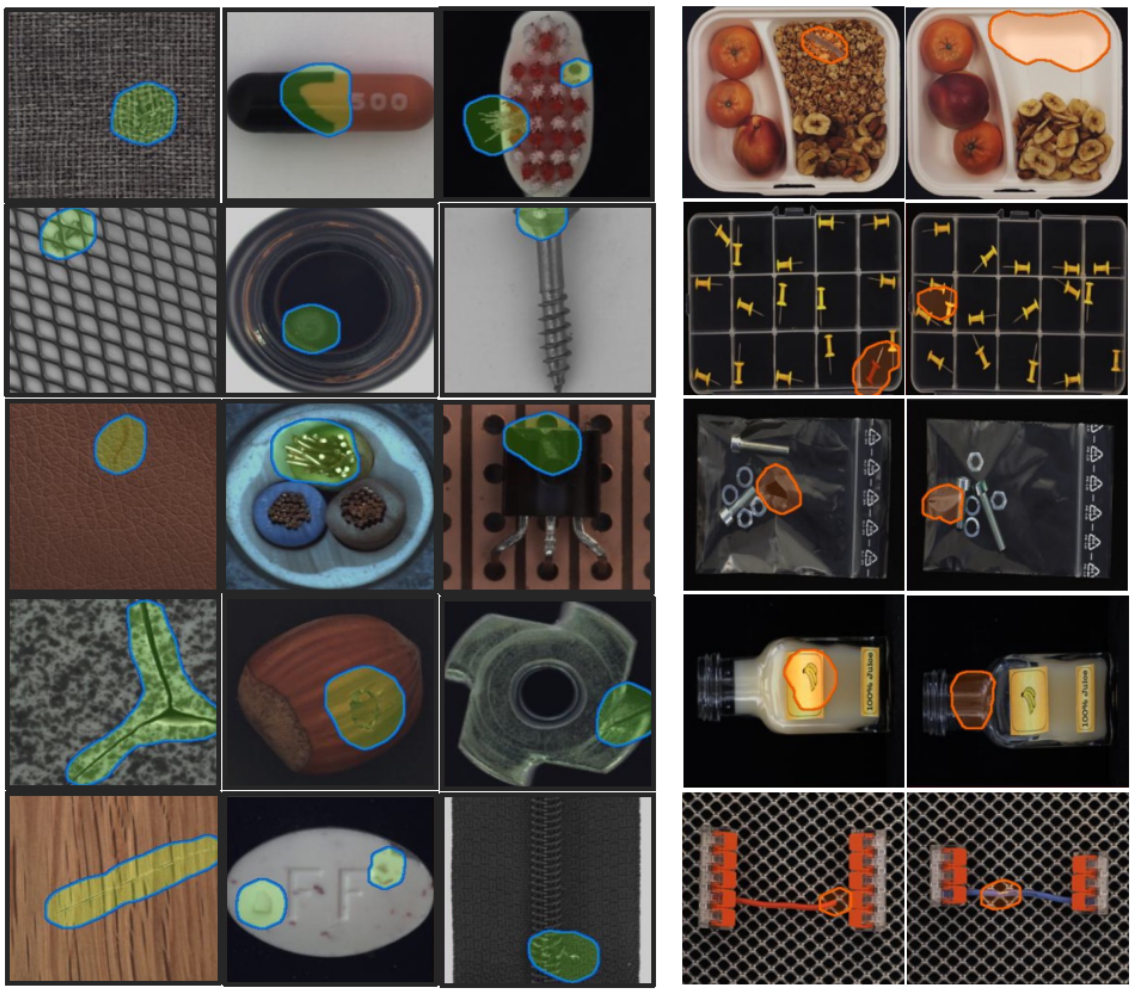

Figure 4.1: Anomaly detection results on the industrial product surface.

pre-trained models for feature extraction and subsequent distribution estimation in anomaly detection.

Nevertheless, detecting anomalies in industrial settings poses distinct challenges. Unlike object detection in natural scenes, where the content is varied and semantically rich, industrial anomalies often manifest in small, irregular areas. This disparity creates a domain bias between natural-scene image detection and industrial anomaly detection, which subsequently reduces the effectiveness of methods based on pre-trained models. To overcome this limitation, various studies [67, 76, 68, 69, 70, 71, 119] have explored self-supervised learning techniques to develop improved feature representations for anomaly detection.

In particular, some approaches [67, 68, 69, 70, 71] focus on simulating natural

defects to enhance representation learning. Nevertheless, these methods often fall short in capturing the full complexity and variability of real defects. Although they attempt to generate defects of varying sizes and locations, the defects are frequently drawn from a fixed-shape distribution, making it easy for DL models to overfit, thus resulting in suboptimal performance. To address this issue, this work proposes a novel defect generation strategy called *DefectMaker*, which produces a diverse array of synthetic defects to enhance representation learning.

On the other hand, industrial images often contain complex information, leading to significant differences between image patches, known as intra-image distribution bias. For instance, in the MVTec AD dataset, the five texture categories exhibit smaller intra-image distribution biases, whereas the ten object categories display larger intra-image distribution biases, as shown in Fig. 4.1. This intra-image distribution bias arises because object categories typically contain more diverse local structures, shapes, and semantic content within a single image, resulting in greater variability among patch-level features. In contrast, texture categories are generally more homogeneous, with repetitive patterns and less variation across patches, leading to a more uniform feature distribution. Such bias poses additional challenges for anomaly detection, as models must distinguish between normal variations within an image and true anomalies. Therefore, effectively modeling and compensating for intra-image distribution bias is crucial for improving the robustness and accuracy of industrial anomaly detection methods.

Therefore, designing an effective distribution estimator is a critical component of representation-based methods, which aim to model patch-level features for anomaly localization and detection. Numerous studies have proposed various solutions to address this challenge. Earlier works have employed Gaussian density estimation (GDE) [50, 51, 67, 52] to model CNN features for anomaly detection. Some approaches have utilized normalizing flows to map image features into a Gaussian distribution. Additionally, several studies [57, 58, 59] have adopted KNN search for feature matching, using the distance as an anomaly score.

KNN-based methods, known for their simplicity and effectiveness, have yielded promising results in anomaly detection. However, most industrial anomaly detection methods rely solely on the vanilla KNN approach, neglecting the local density bias present in feature space. In other domains, existing KNN variants, such as Kth-NN [120], LOF [121], and LDOF [122], consider the local distribution in feature space but show minimal improvement in industrial anomaly detection based on my experiments. This chapter builds on the KNN framework by proposing an LDKNN model that better accounts for density bias in industrial anomaly detection. Our experiments demonstrate that LDKNN outperforms vanilla KNN and other KNN variants, achieving state-of-the-art results.

Overall, to tackle the dual challenges previously identified, this study proposes REB for industrial anomaly detection, comprising two innovative modules: DefectMaker and LDKNN.

DefectMaker characterizes defects by integrating shapes with fills, using an array of both to generate a diverse set of synthetic defects. Notably, DefectMaker employs Bézier curves to craft various defect shapes and leverages a saliency model to guide their placement, resulting in lifelike synthetic defects. These generated defects are subsequently utilized to fine-tune a pre-trained model via a self-supervised learning approach, effectively mitigating domain bias. The refined model is then able to extract more pertinent features for industrial anomaly detection tasks. LDKNN, on the other hand, operates on patch-level features extracted by pre-trained models and performs unsupervised anomaly detection. LDKNN addresses density bias in features by incorporating a local density model that accounts for variations in local feature density. This local density is used to normalize distance measurements during inference, thereby enhancing anomaly detection performance. The proposed framework improves detection accuracy while utilizing fewer parameters, making it suitable for real-time inspection.

The remainder of this chapter is organized as follows: Section 4.2 reviews related work. Section 4.3 details the proposed method. Section 4.4 presents and discusses

the experimental results. Finally, Section 4.5 concludes this chapter.

4.2 Related Works

Anomaly detection (AD) based on learning from normal images is the most prevalent unsupervised approach in defect detection and can be categorized into two main types: construction-based methods and representation-based methods. Recently, the use of pre-trained models on natural image datasets like ImageNet [49] or self-supervised learning tasks has become popular as feature extractors, yielding satisfactory results.

Representation-based methods for anomaly detection create an embedding feature space for normal samples and then estimate or compare these features using various distribution models. These models include parametric approaches such as Gaussian density estimation (GDE) [50, 51, 52] and normalizing flow (NF) [53, 54, 55, 56], as well as non-parametric methods like KNN [57, 58, 59] and kernel density estimation (KDE) [60, 61].

KNN-based anomaly detection techniques typically involve constructing a dictionary of normal features from a pre-trained model, followed by KNN retrieval and distance computation. DN2 [59] was the pioneering method to integrate KNN with a pre-trained CNN feature extractor, yielding promising results in image anomaly detection. Building on this, SPADE [58] extracted patch-level features and utilized KNN for patch-level anomaly detection and localization. PatchCore [57] advanced this approach by creating a patch-level feature dictionary, known as a memory bank, and improving feature representation through multi-level feature extraction. It also leveraged the Coreset algorithm [62] to reduce inference costs. Another method, SMCC [123], introduced an anomaly detection framework based on self-updating memory and center clustering, employing a Gaussian mixture model to fit normal features while simultaneously learning the backbone and updating the memory bank. Similarly, the proposed method also employs a feature memory bank to represent normality. However, the primary distinction between REB and existing memory

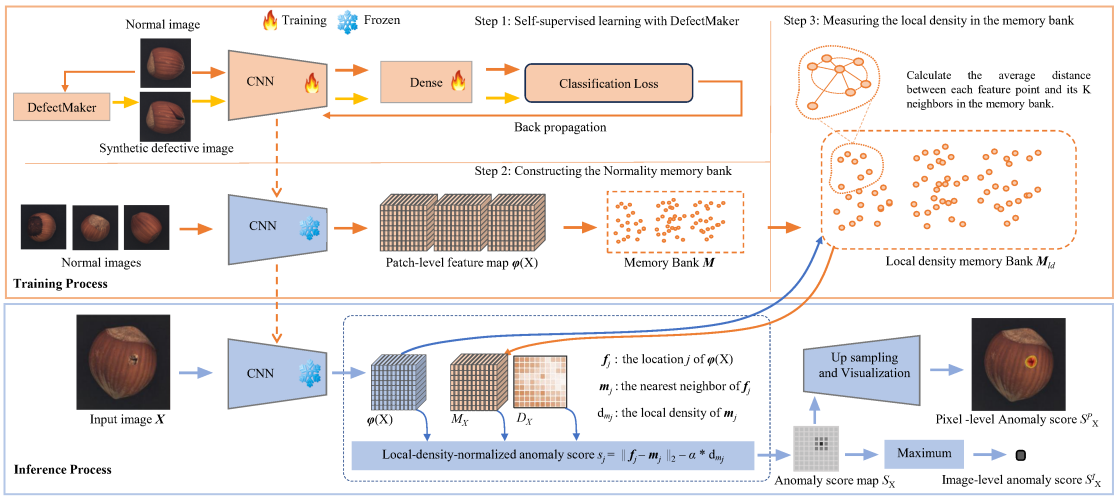

Figure 4.2: Overview of the framework of REB.

bank-based approaches is that REB utilizes a locally density-normalized metric for anomaly detection. Rather than considering the overall dataset density, REB emphasizes the density of data points in the immediate vicinity of the point being analyzed. This concept has been well-established in foundational clustering algorithms [124, 125] and outlier detection [121, 126]. We assume that the memory bank, composed of patch-level features from normal samples, is clean. Therefore, samples with lower local density are less likely to be anomalies, which contrasts with the typical outlier detection assumption. This is achieved by applying a local density model to normalize the distance measurement between the feature under detection and its nearest neighbors.

4.3 Method

This study introduces the REB framework, which operates through two main processes: the training process and the inference process, as illustrated in Fig. 4.2. The training process involves three key steps: 1) self-supervised learning with DefectMaker, 2) constructing the Normality memory bank, and 3) measuring local density in the memory bank.

In the first step, synthetic defective images are generated using the DefectMaker

algorithm, which reduces domain bias and aids in the learning of deep features tailored to industrial applications. In the second step, CNN features extracted from normal images are collected and stored to construct a memory bank, which serves as the representation of normality. In the third step, the local density of each feature point within the memory bank is calculated.

During the inference phase, deep features from test images are extracted using the frozen CNN model. These features are then matched with their nearest neighbors from the memory bank, and an anomaly score is calculated based on the local density-normalized distance.

We summarize the process of measuring local density within the memory bank and the LDKNN inference process as the LDKNN algorithm.

4.3.1 Self-Supervised Learning with DefectMaker

DefectMaker generates a large volume of synthetic defect samples, and the model is trained to distinguish these synthetic images from normal ones through a proxy self-supervised learning (SSL) task.

DefectMaker

DefectMaker is designed to produce various types of defects, including noise, masking, distortion, and other defect types. To fully utilize normal samples and create a diverse range of synthetic defects that more closely resemble natural anomalies, DefectMaker models a defect as a combination of its fill and shape, utilizing a defect-fusing strategy. As depicted in Fig. 4.3, the DefectMaker pipeline consists of three main steps: 1) generating defect shapes; 2) generating defect fills; and 3) synthesizing defect images.

Bézier shape. The rectangles used in previous methods like [67] are overly simplistic, resulting in CNNs trained with these shapes exhibiting limited generalization capabilities. To enhance the realism of the simulated defects, the Bézier curve [127] is employed to generate a variety of shapes. A Bézier curve is a parametric curve commonly used in computer graphics to simulate real-world curves using a set of

discrete 'control points,' rather than relying on a direct mathematical equation.

Two derivative shapes are defined within this approach: Bézier-scar and Bézier-clump shapes. Scar augmentation, as introduced in Cutout [72], utilized a long, thin rectangle to simulate minor defects. In contrast, a Bézier-scar shape is defined as a curved shape generated by the Bézier algorithm, bounded by a rectangular scar, making the defects appear more realistic and similar to actual scratches. The "Bézier-clump shape" is created by generating a curve using the Bézier algorithm, which is then eroded and divided into a cluster of defects.

Defect fill. DefectMaker generates fills with two different distributions. The first type is random noise fill, where noise is added to digital signals or images to introduce randomness or variability—a common data augmentation technique in machine learning tasks. In this study, the random noise fill is generated by controlling properties such as the mean and fluctuation range. The second type is CutPaste fill [67], which involves cutting a region from another image within the same dataset to use as the fill. This technique increases the complexity of the learning task, thereby enhancing the model's discriminative ability.

Defect Fusing. DefectMaker leverages an saliency method [128] to **detect** prominent regions in images, specifically those containing key objects, which are then used to define the defect area. Saliency models are designed to identify visually striking objects or separate the foreground from the background in images using an unsupervised method. In this study, the EDN model [129] is utilized as the saliency extractor. For texture images, the entire image area is treated as a saliency region by default.

The fusion of synthetic defects into an image is a critical step. DefectMaker adopts two fusing styles: Pasting and Blending. Pasting involves directly inserting the defect into the targeted region, which can create a distinct boundary between the defect and the background. Blending, on the other hand, merges the defect with the background using a weighted combination [73], resulting in a more seamless integration between the defect and the surrounding area. By employing both fusing

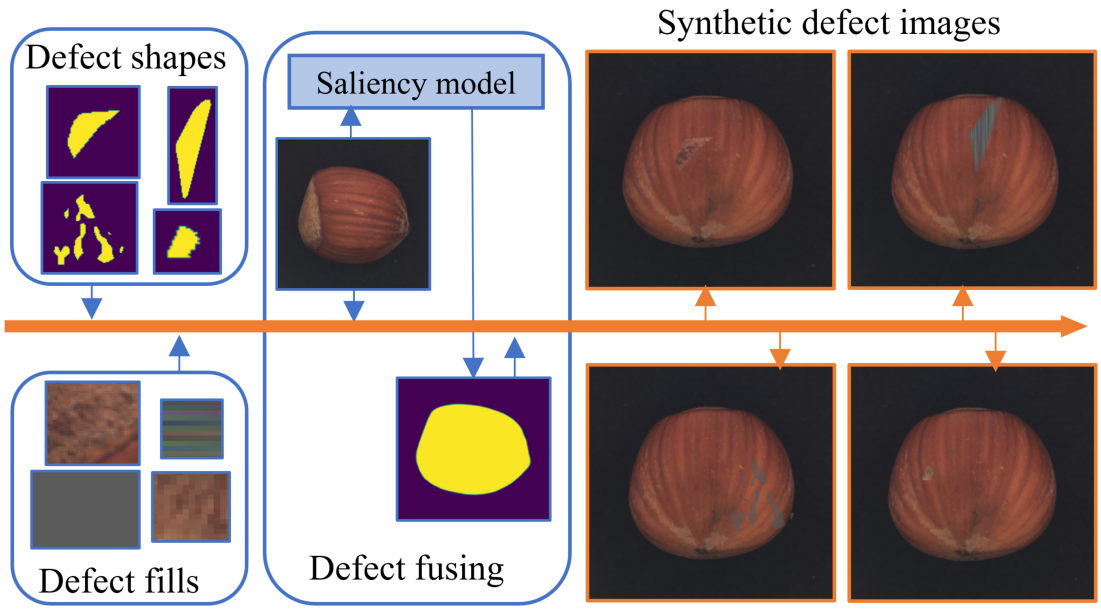

Figure 4.3: DefectMaker Pipline.

techniques simultaneously, DefectMaker enhances the diversity of synthetic samples.

Self-supervised learning (SSL) Task

DefectMaker is capable of generating a vast range of defects with varying sizes, shapes, textures, positions, and appearances. To fine-tune the pre-trained CNN models on ImageNet, this chapter introduces an SSL task that involves learning from these synthetic images. Moreover, defects generated with different configurations in DefectMaker (such as shapes and fills) can be categorized as distinct classes. To fully exploit the diversity of synthetic images, a multi-class classification training strategy is employed. The learning objective for this task is defined using the commonly-used cross-entropy function, as follows:

$$L = -\sum_{c=1}^{C} Y_c \log\left(p\left(Y_c \mid X_i, \Phi\right)\right) \tag{4.1}$$

where X_i is a normal image, Y_c is the label for class c, $p\left(Y_c \mid X_i, \Phi\right)$ is the predicted probability of class c given the input image X_i and the feature extractor Φ, and C is the total number of classes. The model is trained to minimize this loss function, thereby improving its ability to distinguish between normal and defective images. Different configurations of DefectMaker can be used to generate a diverse set

of synthetic defects, which can be treated as different classes. This approach allows the model to learn from a wide range of defect types and enhances its generalization capabilities.

4.3.2 Constructing the Normality Memory Bank

After SSL learning, the second step of the training process involves using the backbone CNN layers to extract patch-level features from normal images and, following the methods described in [58] and [57], constructing a memory bank as the normality representation with these features. In this work, ResNet [130, 131] is employed as the feature extractor. Let X denote a normal image, and $\varphi_j(X) \in \mathbb{R}^{C^j \times H^j \times W^j}$ ($j \in \{1, 2, 3, 4\}$) represent the feature map of X at the j-th hierarchy, where C^j, H^j, and W^j are the feature channel, height, and width, respectively.

A common strategy, as used in [58, 67, 59], is to take φ_4, the feature map from the last hierarchy of ResNet18. However, this approach may not be ideal for industrial anomaly detection because the deeper features from the last hierarchy are often biased towards the proxy task. Given that anomalies in industrial images may be characterized by small spatial sizes and irregularities, these features can be lost as the network becomes deeper. Therefore, this work, following [57], adopts the second and third hierarchies $\varphi_2(X)$ and $\varphi_3(X)$ for X.

To maximize the utility of the multi-hierarchy feature maps, a feature aggregation module is introduced to combine features from multiple hierarchies. Specifically, the aggregation process involves two steps: local spatial aggregation and multi-hierarchy aggregation. First, a local average pooling operation is applied to $\varphi_2(X)$ and $\varphi_3(X)$ to obtain a larger receptive field, followed by upsampling $\varphi_3(X)$ to match the resolution of $\varphi_2(X)$. Then, $\varphi_2(X)$ and $\varphi_3(X)$ are aggregated along the channel dimension into an integrated feature map, denoted as $\varphi(X) = \varphi_2(X) \oplus \varphi_3(X) \in \mathbb{R}^{C' \times H' \times W'}$, where C', H', and W' are equal to $C^2 + C^3$, H^2, and W^2, respectively. A location $\varphi(X, h, w) \in \mathbb{R}^{C'}$ represents a feature vector associated with a patch in the image. Finally, the feature vectors of the normal

training images are collected into a feature dictionary (memory bank $\boldsymbol{M}$).

Algorithm 1: LDKNN algorithm

Training Process:

Input: Patch-level Memory Bank $\boldsymbol{M}$

$\mathbf{M}_{ld} = \{emptyset\}$

For $\mathbf{m}_i \in \boldsymbol{M}$ do

retrieve the KNN $\boldsymbol{m}_i^1, \boldsymbol{m}_i^2, ..., \boldsymbol{m}_i^K$

$d_{m_i} = mean_K(||\boldsymbol{m}_i - \boldsymbol{m}_{ik}||_2), 1 \leq k \leq K$

$\boldsymbol{M}_{ld} \leftarrow \boldsymbol{M}_{ld} \cup \{(\boldsymbol{m}_i, d_{m_i})\}$

Return Memory bank with local density $\boldsymbol{M}_{ld}$

Inference Process:

Input: Input image X

Patch-level anomaly score map $\boldsymbol{S}_X = \{emptyset\}$

Patch-level feature map $\varphi(X) = \{\boldsymbol{f}_j\}, 1 \leq j \leq \grave{H} \times \grave{W}$

$(\boldsymbol{m}_j, d_{m_j}) = \underset{(\boldsymbol{m}_i, d_{m_i}) \in \boldsymbol{M}_{ld}}{argmin} ||\boldsymbol{f}_j - \boldsymbol{m}_i||_2$

$s_{f_j} = ||\boldsymbol{f}_j - \boldsymbol{m}_j||_2 - \alpha * d_{m_j}$

$\mathbf{S}_X \leftarrow \boldsymbol{S}_X \cup s_{f_j}$

Image-level anomaly score: $S_X^I = max(\boldsymbol{S}_X)$

Return $\boldsymbol{S}_X^I, S_X$

4.3.3 Local density measurement in the memory bank M

The vanilla KNN-based anomaly detection process involves retrieving the K most similar feature points for a query feature and computing the average Euclidean distance as the anomaly score. However, the vanilla KNN approach assumes that the patch-level feature space is homogeneous, which is an overly simplistic assumption. This method does not account for the local density bias caused by the complex intra-image distribution and the irregular characteristics of defects.

In the final training step, local density measurements are performed for each feature point in the memory bank. Specifically, for each feature point d_{m_i}, its K nearest neighbor features (KNNs) are retrieved, and the average distance between the feature point and its KNNs is calculated as the local density using the following equation:

$$d_{m_i} = mean_K \left(||\boldsymbol{m}_i - \boldsymbol{m}_i^k||_2 \right), \quad 1 \leq k \leq K, \tag{4.2}$$

where m_i^k denotes the k-th neighbor of m_i, and $mean_K(\cdot)$ represents the mean function, calculating the average Euclidean distance between m_i and its K nearest

neighbors.

This distance serves as a measure of the local density in the memory bank. The smaller the distance, the higher the density. As a result, a local density-based memory bank $\boldsymbol{M}_{ld}$ is constructed, where each feature point is assigned a local density attribute value.

4.3.4 LDKNN inference process for anomaly detection

After the training process, two core components are obtained: a CNN feature extractor and a local density memory bank $\boldsymbol{M}_{ld}$ for unsupervised anomaly detection. In the inference process in Fig. 4.2, for each testing image X, each testing image X is first fed into the CNN feature extractor to obtain the aggregated feature map $\varphi(X)$. For each feature point $\boldsymbol{f}_j$ in $\varphi(X)$,1-NN is used to retrieve the most similar feature in the memory bank, and the Euclidean distance between them is determined as the anomaly score. Further, the local-density-normalized anomaly score $s_{\boldsymbol{f}_j}$ for patch feature $\boldsymbol{f}_j$ is calculated as

$$s_{f_j} = ||\boldsymbol{f}_j - \boldsymbol{m}_j||_2 - \alpha * d_{m_j}, \tag{4.3}$$

where $(\boldsymbol{m}_j, d_{m_j})$ is the nearest neighbor of f_j in the memory bank M_{ld},

$$(\boldsymbol{m}_j, d_{m_j}) = \underset{(\boldsymbol{m}_i, d_{m_i}) \in M_{ld}}{argmin} ||\boldsymbol{f}_j - \mathbf{m}_i||_2. \tag{4.4}$$

The anomaly score is normalized using d_{m_j}, with α serving as the local density coefficient, reflecting the degree of density bias between dense and sparse areas. Setting an appropriate value for α is crucial. Traversing each feature point in the feature map and calculating its anomaly score yields a patch-level anomaly score map S_X. The image-level anomaly score S_X^I for an image X is calculated as the maximum of the patch anomaly score map $\boldsymbol{S}_X$ as

$$S_X^I = max(\boldsymbol{S}_X), x_i \in X. \tag{4.5}$$

REB is not only good at image-level anomaly detection results and is also capable of achieving pixel-level anomaly detection results $\boldsymbol{S}_X^P$ by upsampling $\boldsymbol{S}_X$. We summarize the measuring of the local density in the memory bank process and local-density normalized inference process as the LDKNN algorithm, as shown in Algorithm 1.

	KNN	LDKNN
Training Time Complexity	$O(1)$	$O(n^2)$
Inference Time Complexity	$O(n \cdot d)$	$O(n \cdot d)$
Space Complexity	$O(n \cdot d)$	$O(n \cdot d)$

Table 4.1: Computational Complexity Comparison between vanilla KNN and LDKNN.

4.3.5 Computational complexity of LDKNN

This section analyzes the complexity of LDKNN and compare it with the vanilla KNN method used by PatchCore, as summarized in Table 4.3.4.

The training time complexity of vanilla KNN is $O(1)$, since it merely involves storing data without an explicit learning phase. The inference time complexity is $O(n \cdot d)$, as it requires a full scan of the dataset for each query, where n represents the number of data points and d denotes the dimension of the feature space. The space complexity is also $O(n \cdot d)$, corresponding to the storage needed for the entire dataset.

LDKNN enhances the vanilla KNN approach by incorporating local density calculations for each data point during the training phase. This involves determining the average distance between a feature point and its K nearest neighbors to quantify the local density. Assuming a linear search is used, the training time complexity of LDKNN is $O(n^2)$. During inference, LDKNN extends vanilla KNN by integrating local density into the anomaly scoring process. Despite this additional step, the inference time complexity remains $O(n \cdot d)$, similar to KNN, because the local density adjustment is a constant-time operation. The space complexity of LDKNN is

equivalent to that of KNN at $O(n \cdot d)$, as it requires storage of the entire training set along with an additional local density value for each data point.

Overall, LDKNN is highly efficient, improving anomaly detection performance by learning local density without increasing the inference time complexity or space complexity.

4.4 Experiments

We conducted a series of experiments on three datasets to evaluate the effectiveness of REB. In this section, REB is defined as the combination of DefectMaker fine-tuned weights and LDKNN detection. The most relevant work for comparison is PatchCore, which combines ImageNet pre-trained weights with vanilla KNN detection.

To facilitate a more detailed comparison of each module's effects, this section introduces the following baselines: the combination of DefectMaker fine-tuned weights with vanilla KNN detection, referred to as DefectMaker, and the combination of ImageNet pre-trained weights with LDKNN detection, referred to as LDKNN.

4.4.1 Datasets and Metrics

MVTec AD dataset is a dataset for unsupervised anomaly detection with 15 categories, including ten objects and five textures. There are 5,354 industrial product images, where 3,629 normal images are for training and validation and 1,725 for testing. The training set only contains normal images, while the testing set contains images of various types of defects and normal defects.

MVTec LOCO AD dataset [106] is another dataset for industrial anomaly detection. It consists of 1,772 images for training, 304 for validation, and 1,568 for testing. The training and testing sets are used in the experiment. Different from the MVTec AD dataset, it demonstrates five more complex inspection scenarios. Each possesses two difficult anomaly categories: logical and structural anomalies.

BTAD dataset [107] is another dataset similar to MVTec AD, which is composed of RGB images representing three distinct industrial products and consists of 1,799 images for training and 741 images for testing.

Metrics. In the field of unsupervised anomaly detection, model evaluation typically focuses on two main aspects: anomaly detection performance and anomaly localization capability.

Anomaly detection performance refers to the model's ability to determine whether an entire image contains anomalies. This requires the model to classify the entire image as either normal or anomalous. Anomaly localization capability, on the other hand, assesses the model's ability to identify specific anomalous pixels within an image, requiring a binary classification (anomalous or not) for each pixel.

For classification tasks, the Receiver Operating Characteristic (ROC) curve [132] is a commonly used graphical tool that displays a classification model's performance across all possible thresholds. The ROC curve is plotted by comparing the True Positive Rate (TPR) against the False Positive Rate (FPR). The Area Under the ROC Curve (AUROC) [133] represents the overall classification performance of a binary classifier model across different thresholds.

Based on these metrics, Image-level AUROC (Im.AUROC) and Pixel-level AUROC (Pi.AUROC) are used to evaluate the anomaly detection performance and anomaly localization performance, respectively.

4.4.2 Training Details and Hyperparameter Setting

DefectMaker. We combined 3 different defect shapes with 2 types of defect fills to generate 6 unique defect categories. Additionally, this study incorporated a non-defect class, framing the DefectMaker self-supervised proxy task as a 7-class classification problem. The SGD optimization algorithm was utilized with an initial learning rate of 0.03, which gradually decreased according to a cosine schedule. The input images were resized to 256×256 pixels, and a batch size of 1024 was used. The model was trained for 300 iterations. Due to memory limitations, this section

implemented a training strategy that involved performing multiple forward passes before executing a single backward pass.

LDKNN. Regarding the two parameters of LDKNN, K and the local density (LD) coefficient, the LD coefficient was set to 1 across all subsequent experiments on three datasets unless otherwise specified. For each dataset, this study applied the same K value across all categories. However, because the number of training images per class varies in the MVTec AD and MVTec LOCO AD datasets, the optimal K parameter for achieving the best results may differ across categories. It may be interesting to assign a variable K for each class. As shown in Table 4.4.2 and 4.4.4, this section presented the results using both a constant K and a variable K across different classes.

Method	Backbone	Pi.AUROC	Im.AUROC														
		15 Total	carpet	grid	leather	tile	wood	bottle	cable	capsule	hazelnut	metal_nut	pill	screw	toothbrush	transistor	zipper	15 Total
FYD [52]	Res18	97.4	98.3	97.4	100	95.4	98.2	100	94.3	93.2	99.8	99	94.9	89.7	99.9	97.2	96.8	97.3
FastFlow [54]		97.2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	97.9
Cflow [53]		**98.1**	98.5	96.8	98.6	99	94	98.8	99.5	97.6	98.3	97.4	95.1	98.4	92.7	93.5	97.7	96.75
NSA [76]		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	97.2
DifferNet [55]		-	92.9	84	88	99.4	99.8	99	95.9	86.9	99.3	96.1	88.8	96.3	91.9	91.1	88.6	94.9
CutPaste [67]		96.0	93.1	99.9	100	93.4	98.6	98.3	80.6	96.2	97.3	99.3	92.4	86.3	98.3	95.5	99.4	95.2
PatchCore [57]		97.4	99.4	97.7	100.0	98.9	98.9	100.0	97.7	97.9	100.0	99.6	90.5	98.1	100.0	96.5	97.6	98.2
SMCC [123]		97.9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	97.2
SimpleNet [119]		95.7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	98.3
RD++ [134]		97.6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	98.6
DeSTSeg [135]		97.9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	98.6
(Our) DefectMaker		97.7	98.9	99.7	100.0	100.0	99.3	100.0	99.0	95.1	99.3	100.0	97.3	97.5	100.0	95.2	100.0	98.7
(Our) LDKNN (K=5)		97.9	99.5	99.2	100.0	99.8	99.0	100.0	99.8	98.8	100.0	99.8	95.2	97.4	99.4	99.3	99.2	**99.1**
(Our) LDKNN (Variable K)		98.0	99.6	99.6	100.0	99.9	99.0	100.0	99.9	99.0	100.0	99.9	97.0	98.1	100.0	99.8	99.6	**99.4**
(Our) REB (K=5)		98.0	98.9	99.6	100.0	100.0	99.4	100.0	99.5	97.9	99.4	100.0	98.7	98.1	98.9	99.6	100.0	**99.3**
(Our) REB (Variable K)		98.0	99.0	99.7	100.0	100.0	99.5	100.0	99.7	98.3	99.5	100.0	99.3	98.3	100.0	99.7	100.0	**99.5**
PatchCore [57]	WR50	**98.4**	98.4	99.3	100.0	99.8	99.2	100.0	99.5	98.9	100.0	100.0	94.6	97.0	99.7	99.4	99.7	99.0
SMCC [123]		98.3	100	99.3	100	100	99.6	100	96.6	96.6	100	99.6	95.5	91.1	100	100	98.5	98.5
SimpleNet [119]		98.1	99.7	99.7	100	99.8	100	100	99.9	97.7	100	100	99.0	98.2	99.7	100	99.9	**99.6**
RD++ [134]		98.3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	99.4
ReConPatch [136]		98.3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	99.6
THFR [137]		98.2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	99.2
(Our) DefectMaker		98.1	99.8	100.0	100.0	100.0	99.7	100.0	99.9	98.0	99.5	100.0	98.3	96.0	99.2	96.9	100.0	99.1
(Our) LDKNN (K=9)		**98.4**	98.1	99.2	100.0	99.9	99.5	100.0	100.0	99.8	100.0	100.0	97.5	94.6	97.8	100.0	99.9	99.1
(Our) LDKNN (Variable K)		**98.4**	98.4	99.3	100.0	100.0	99.5	100.0	100.0	100.0	100.0	100.0	99.3	97.0	99.7	100.0	100.0	99.5
(Our) REB (K=9)		98.3	99.6	100.0	100.0	100.0	99.7	100.0	100.0	99.3	99.7	100.0	99.0	94.7	96.1	99.9	100.0	99.2
(Our) REB (Variable K)		**98.4**	99.8	100.0	100.0	100.0	99.7	100.0	100.0	99.3	99.7	100.0	99.5	96.0	99.2	100.0	100.0	99.5

Table 4.2: Anomaly detection performance (Im.AUROC and Pi.AUROC) using backbones Resnet18 and WideResnet50 on MVTec AD dataset. The best results are shown in bold.

4.4.3 Anomaly Detection on MVTec AD

In this work, REB is evaluated using ResNet18 (Res18) and WideResNet50 (WR50) as backbones, primarily to ensure a fair comparison with related representation-based methods. The results indicate that the problem has been adequately addressed using smaller networks, thus negating the need for larger models.

In Table 4.4.2, the AUROC scores for various methods on the MVTec AD dataset are presented. The last column shows the average results across all 15 categories. The Pi.AUROC scores are detailed in the third column. With Res18 as the backbone, an Im.AUROC of 98.7% is achieved by DefectMaker alone, which surpasses other representation-based methods. Specifically, when comparing DefectMaker, which utilizes a basic KNN approach for detection, to PatchCore, an average increase of 0.5% in Im.AUROC across 15 different classes is observed. This improvement highlights the successful reduction of domain bias and the enhancement of feature representation. An Im.AUROC of 99.1% is recorded by LDKNN alone, which also outperforms other representation-based methods. When LDKNN is compared with PatchCore, which similarly relies on ImageNet pre-training for representation, an average increase of 0.9% in Im.AUROC across 15 classes is noted, suggesting significant improvement in feature representation.

The best results are obtained by employing REB (DefectMaker and LDKNN), with Im.AUROC scores of 99.3% (K=5) and 99.5% (Variable K). Even with WR50 as the backbone, where PatchCore already achieves a 99.1% Im.AUROC, marginal improvements are still achieved by the approaches—DefectMaker, LDKNN, and REB. A comparative analysis between Res18 and WR50 backbones indicates that when REB is implemented on Res18, results comparable to those of WR50 are delivered. This suggests that a balance between fewer parameters and faster detection speeds is achieved by the proposed REB.

Among other methods, SimpleNet [119] and ReConPatch [136] achieve an Im.AUROC of 99.6% using the WR50 backbone. In contrast, exceptional performance is demonstrated by the methods with the ResNet18 backbone. For example, Im.AUROC scores of 98.7%, 99.1%, and 99.5% are achieved by DefectMaker, LDKNN (K=5), and REB (Variable K), respectively, all outperforming SimpleNet's 98.3% on Res18. The state-of-the-art (SOTA) result of 98.6% on Res18, previously set by RD++ [134] and DeSTSeg [135], is also surpassed by REB. This underscores the efficiency and effectiveness of the approaches, particularly when leveraging the

smaller backbone.

While the primary focus of this section is on detection performance, as quantified by Im.AUROC, it is noteworthy that a slight improvement in localization performance is also observed, as indicated by the Pi.AUROC scores in the third column.

4.4.4 Anomaly Detection on MVTec LOCO AD

The MVTec LOCO AD dataset, introduced by Bergmann et al. [106], presents a significant challenge in anomaly detection. To address this, the GCAD model was introduced, achieving an impressive 83.3% Im.AUROC. This model was rigorously tested against various established methods, including autoencoders (AE) [11], variational autoencoders (VAE) [42], the Variation Model (VM) [138], the memory-guided autoencoder (MNAD) [139], SPADE [58], and the Student–Teacher (ST) model [140]. Subsequently, SINBAD [141] was introduced, which, by employing set features to simulate the distribution of elements in each sample, effectively identified anomalies caused by unusual combinations of normal elements, demonstrating superior results on the MVTec LOCO AD dataset. The Template-guided Hierarchical Feature Restoration (THFR [137]) method further optimized anomaly detection in images by compressing and restoring hierarchical features through bottleneck compression and template-guided compensation, achieving promising results.

In this section, a comparative analysis of the proposed approach against other methods in the field is presented, with a specific focus on PatchCore, the method most similar to ours. Similar to the experiments on the MVTec dataset, the proposed approach was evaluated under various scenarios, including the use of DefectMaker, LDKNN, and REB, and across different backbones: ResNet18 (Res18), WideResNet50 (WR50), and WideResNet101 (WR101). Our analysis considers overall performance while also examining specific categories of structural and logical anomalies.

When Res18 was used as the backbone, DefectMaker alone achieved an Im.AUROC of 77.5, with scores of 83.9 and 71.1 for the structural and logical classes,

respectively. While this performance is inferior to GSAD and SINBAD, it outperforms PatchCore using Res18. Notably, DefectMaker outperforms both methods in the structural class. Further analysis reveals that the self-supervised training provided by DefectMaker effectively improves detection performance in the structural anomaly category, but leads to a decrease in performance for detecting logical anomalies, as DefectMaker is primarily designed to generate structural anomalies.

Moreover, when LDKNN alone was used, an average Im.AUROC of 85.4 (K=45) and 86.1 (Variable K) was achieved, approaching the performance of the state-of-the-art method SINBAD on WR50. This reaffirms the effectiveness of LDKNN in mitigating feature density bias and demonstrates its ability to capture the intricate intra-image distribution. Compared to LDKNN, REB achieves better performance in detecting structural anomalies but performs worse in detecting logical anomalies due to the influence of DefectMaker.

Furthermore, when WR50 was used as the backbone, a superior result of 88.1 Im.AUROC was achieved, surpassing all other methods. Experiments were also conducted using WR101, and only the results of LDKNN are showcased, considering the potential negative impact of DefectMaker on the detection of logical anomalies. As shown in Table 4.4.4, an even better result with an impressive Im.AUROC score of 88.8% was obtained on WR101.

4.4.5 Anomaly Detection on BTAD

Table 4.4 presents a comparative analysis of REB against several state-of-the-art (SOTA) methods using the WR50 backbone on the BTAD dataset, which comprises 2,540 images across three different categories of industrial products. Compared to the SOTA methods (i.e., PaDiM [51], RD++ [134], and ReConPatch [136]), REB demonstrates competitive performance in terms of both Im.AUROC and Pi.AUROC.

While the SOTA methods each excel in specific categories, REB distinguishes itself with an impressive average Im.AUROC score of 96.0%, underscoring its supe-

Method	Backbone	Pi.AUROC	Im.AUROC							
		Total	BreakfastBox	ScrewBag	Pushpins	SplicingConnectors	JuiceBottle	structural	logical	Total
(Baseline) VAE [42]	-		-	-	-	-	-	54.8	53.8	54.3
(Baseline) AE [11]	-		-	-	-	-	-	56.6	58.1	57.3
(Baseline) VM [138]	-		-	-	-	-	-	58.9	56.5	57.7
(Baseline) f-AnoGAN [142]	-		-	-	-	-	-	62.7	65.8	64.2
(Baseline) MNAD [139]	-		-	-	-	-	-	70.2	60.0	65.1
(Baseline) SPADE [58]	-		-	-	-	-	-	66.8	70.9	68.9
(Baseline) ST [140]	-	-	-	-	-	-	-	88.3	66.4	77.3
GCAD [106]	-		-	-	-	-	-	80.6	86.0	83.3
PatchCore [57]	Res18	72.4	71.8	93.6	68.1	65.0	77.3	80.5	71.7	76.1
PatchCore [57]	WR50	74.3	74.1	93.4	69.2	66.4	82.2	83.6	72.9	78.3
PatchCore [57]	WR101	**76.6**	80.6	95.5	71.0	68.5	84.7	85.3	76.5	80.9
THFR [137]	WR50		-	-	-	-	-	-	-	86.0
SINBAD [141]	WR50	-	77.5	92.2	71.2	66.9	80.6	84.7	**88.9**	86.8
(Our) DefectMaker		70.8	75.4	91.8	72.6	62.4	80.1	83.9	71.1	77.5
(Our) LDKNN (K=45)		68.5	87.9	96.2	83.4	72.4	82.8	**90.1**	80.7	85.4
(Our) LDKNN (Variable K)	Res18	70.28	88.2	97.1	84.0	72.4	84.2	**90.9**	81.3	86.1
(Our) REB (K=45)		67.7	89.7	96.9	78.1	75.5	84.0	**91.3**	80.0	85.7
(Our) REB (Variable K)		68.4	89.7	97.3	78.2	75.6	84.3	**91.4**	80.3	85.9
(Our) DefectMaker		72.8	77.5	92.2	71.2	66.9	80.6	86.9	71.1	79.0
(Our) LDKNN (K=45)		70.0	90.6	96.9	87.0	71.1	88.7	**92.6**	83.0	**87.8**
(Our) LDKNN (Variable K)	WR50	70.7	90.6	97.6	87.2	71.6	89.0	**92.7**	83.5	**88.1**
(Our) REB (K=45)		68.0	91.0	96.5	79.2	75.0	88.5	**93.3**	80.9	**87.1**
(Our) REB		68.52	91.1	97.3	79.6	75.3	88.9	**93.2**	81.6	**87.4**
(Our) LDKNN (K=45)	WR101	74.4	91.0	98.9	88.8	70.1	90.1	**91.4**	85.8	**88.6**
(Our) LKDKN (Variable K)		73.5	91.1	98.9	88.9	70.9	90.2	**91.5**	86.1	**88.8**

Table 4.3: Anomaly detection performance Im.AUROC and Pi.AUROC on MVTec LOCO AD datasets. The best results are shown in bold.

Method	Class 01	Class 02	Class 03	Mean
VT-ADL [107]	(97.6, **99.0**)	(71.0, 94.0)	(82.6, 77.0)	(83.7, 90.0)
SPADE [58]	(91.4, 97.3)	(71.4, 94.4)	(99.9, 99.1)	(87.6, 96.9)
PaDiM [51]	(98, 97.0)	(82.0, 96.0)	(99.4, 98.0)	(93.7, 97.3)
FastFlow [54]	(99.4, 97.1)	(82.4, 93.6)	(91.1, 98.3)	(91.0, 96.3)
PyramidFlow [143]	(**100**, 97.4)	(88.2, **97.6**)	(99.3, 98.1)	(95.8, 97.7)
RD++ [134]	(96.8, 96.2)	(**90.1**, 96.4)	(**100**, 99.7)	(95.6, 97.4)
PatchCore [57]	(98, 96.9)	(81.6, 95.8)	(99.8, 99.1)	(93.1, 97.3)
ReconPatch [136]	(99.7, 96.8)	(87.7, 96.6)	(**100**, 99.0)	(95.8, **97.5**)
(Our) REB (K=57)	(99.6, 94.7)	(88.5, 95.6)	(99.8, 99.7)	(**96.0**, 97.2)

Table 4.4: Comparison of state-of-the-art models on the BTAD dataset with REB, showing (Im.AUROC, Pi.AUROC) for each class. The best results are shown in bold.

rior capability in detecting image-level anomalies. Furthermore, REB demonstrates its ability to accurately pinpoint anomalies at the pixel level, achieving an average Pi.AUROC of 97.2%.

4.4.6 Various Feature Representations

This section explores the performance of the proposed method using various feature representations derived from pre-trained ImageNet, DRAEM, CutPaste, and DefectMaker with different defect configurations. The ImageNet representation refers to directly using pre-trained weights from the ImageNet dataset, while the other repre-

sentations involve fine-tuning ImageNet using self-supervised tasks to reduce domain bias. For a fair comparison, the same backbone (Res18) and training strategy were used during fine-tuning, with the only difference being the method of generating synthetic defects. These different representations were evaluated across three different anomaly detection methods: LDKNN, KNN, and GDE, as shown in Table 4.5.

Following convention, the GDE method utilized features from the last layer (φ_4) of Res18, whereas LDKNN and KNN utilized features from shallower layers (φ_2 and φ_3). The performance of deeper versus shallower features can be assessed by separately evaluating the results of GDE and LDKNN. Notably, LDKNN achieved better performance even with shallower features on the MVTec AD dataset.

CutPaste employs rectangular shapes (Rect and RectScar) as anomaly shapes, and DRAEM uses Perlin noise to generate anomaly shapes, while DefectMaker uses more diverse Bézier shapes. In addition to the CutPaste fill, DefectMaker incorporates random noise fill to increase the diversity of synthetic defect images.

The comparison of results clearly shows that DefectMaker generates more effective synthetic defects and produces better feature representations. Additionally, it is evident that the performance disparity of the GDE method becomes increasingly significant with varying representations, which reinforces the understanding that deeper network layers exacerbate domain bias. PatchCore tackles this challenge by opting for shallow features to construct its memory bank, effectively sidestepping the domain bias that occurs in the deeper layers.

Representation	Defect Shape				Defect Fill			Im.AUROC		
Strategy	Rect	RectScar	Bézier	Perlin	CutPaste	Random Noise	External dataset	GDE on φ_4	KNN on $\varphi_2+\varphi_3$	LDKNN on $\varphi_2+\varphi_3$
ImageNet								90.3	98.2	99.1
DRAEM				✓			✓	91.0	98.6	98.9
CutPaste	✓	✓			✓			94.5	98.2	98.8
DefectMaker			✓		✓			**97.7**	**98.8**	**99.4**
DefectMaker			✓			✓		**96.6**	98.4	99.1
DefectMaker			✓		✓	✓		**97.3**	**98.7**	**99.3**

Table 4.5: Image-level anomaly detection performance (Im.AUROC) on different feature representations on MVTec AD.

Dataset	Representation	KNN	Kth-NN	LOF	LDOF	LDKNN (Our)
MVTec LOCO	ImageNet	75.1	74.5	78.3	78.1	**83.1**
	DefectMaker	75.7	75.8	78.7	79.8	**84.2**
MVTec	ImageNet	98.2	97.3	97.2	96.2	**98.8**
	DefectMaker	98.7	98.6	96.6	95.8	**99.3**

Table 4.6: Evaluating KNN Methods for Anomaly Detection on Im.AUROC scores. The best results are shown in bold.

4.4.7 Various KNN Variants

In addition to the standard KNN, this section also compared REB's Im.AUROC performance against various KNN variants, including Kth-NN, LOF, and LDOF. For all KNN variants, the neighbor size K was varied within the set $\{3, 5, 7, 9, 11, 13, 15\}$, and the best results are reported in Table 4.4.6. Additionally, the other hyperparameter, the LD coefficient α, was fixed at 1 for a fair comparison. As shown in Table 4.4.6, the proposed LDKNN outperforms the other KNN variants on the two datasets, regardless of whether the DefectMaker self-supervised learning is used. This superior performance is due to LDKNN's combination of distance and density metrics for anomaly detection, with the LD coefficient balancing the two. In contrast, distance-based KNN and Kth-NN overlook the complex intra-image distribution. LOF and LDOF, which are density-based, show uneven detection ability in different density regions, being more sensitive in low-density areas but less effective in high-density areas.

4.4.8 Coreset Algorithm and Inference Time

Detection efficiency is crucial for industrial manufacturing, as it directly affects the production cost and efficiency. The inference time of KNN-based anomaly detection methods typically scales with the size of the backbone and the memory bank. This issue has been previously discussed by works such as [58] and PatchCore [57]. PatchCore addresses this challenge by incorporating a Coreset algorithm [144], which selects a subset of the memory bank to approximate the full dataset, thereby reducing computational costs.

This section integrates the Coreset algorithm with the proposed REB framework

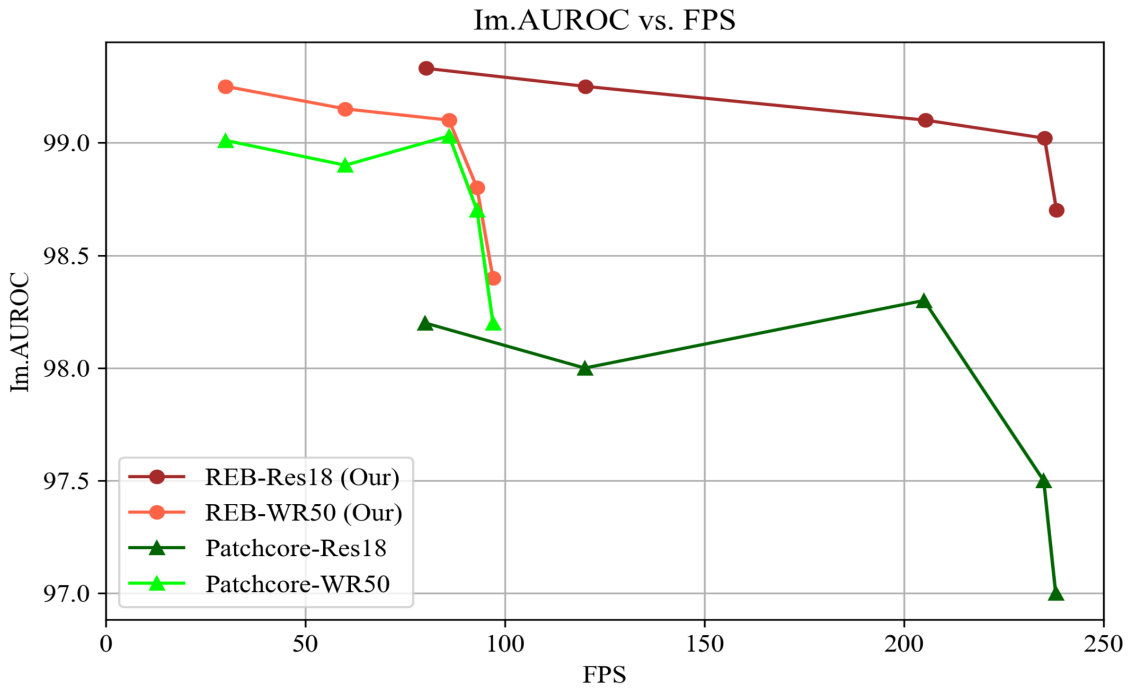

Figure 4.4: Im.AUROC vs. FPS (frames per second)

and compare its performance with PatchCore. Following convention, REB-n% is used to denote the percentage of the Coreset used. The Coreset proportions tested were 100%, 50%, 10%, 1%, and 0.1%. All experiments were conducted on the same RTX3090 GPU to ensure a fair comparison. We used Res18 and WR50 as the backbones for REB and PatchCore, respectively.

Figure 4.4 illustrates the relationship between FPS (frames per second) and Im.AUROC performance, with the x-axis representing FPS and the y-axis representing Im.AUROC. The results demonstrate that REB consistently outperforms PatchCore across all Coreset proportions, particularly when using smaller backbone networks. When the memory bank size is relatively large, REB achieves better performance by more effectively accounting for the local distribution of the feature space.

Focusing solely on the FPS (frames per second), the comparison reveals that PatchCore and REB achieve nearly identical FPS rates when using the same backbone and Coreset ratio. This similarity is primarily due to the fact that the only structural difference between PatchCore and REB during the inference phase is the

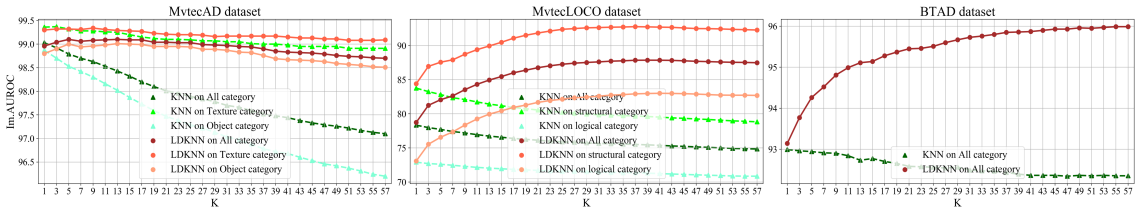

Figure 4.5: Evaluating the Influence of K on Im.AUROC in LDKNN Algorithm, with KNN as the Baseline Method for Comparison.

use of KNN in PatchCore and LDKNN in REB. Since LDKNN only adds a simple constant value calculation during inference, the time difference between LDKNN and KNN is negligible—less than 0.1 ms, according to my measurements. This data further confirms that LDKNN and vanilla KNN share similar inference time complexities.

4.4.9 Evaluation on Various Hyper-parameters K

This section explored the effect of varying K from 1 to 57 on the average Image-Level AUROC across three datasets: MVTec AD, MVTec LOCO AD, and BTAD. WR50 was primarily used as the backbone. Since KNN detection with the WR50 backbone has already yielded promising results on MVTec AD, simply examining the impact of changes in K was considered insufficient. Therefore, Res18 was employed as the backbone for MVTec AD. Additionally, not only was the average performance across all 15 classes of the MVTec AD dataset analyzed, but detailed insights into the 5 texture and 10 object classes were also provided. For the MVTec LOCO AD dataset, this section further categorized the results into structural and logical anomaly classes.

As shown in Figure 4.5, LDKNN produces results similar to KNN when K is set to 1. However, as K increases, LDKNN clearly distinguishes itself, consistently improving in both anomaly detection and localization, in contrast to KNN, which shows a notable decline in performance as K grows. This difference is likely due to KNN's tendency to assign disproportionately low anomaly scores to high-density regions and higher scores to sparse areas, thus exacerbating imbalance. Based on the

results from these three experiments, LDKNN, compared to KNN, more effectively accounts for the distribution of the feature space, effectively reducing local density bias.

The value of K in the LDKNN algorithm is pivotal in determining the number of nearest neighbors considered when calculating the local density of each feature point within the memory bank. This K value directly influences the measurement of local densities and, consequently, the normalization of anomaly scores during the detection process. In the context of LDKNN, K affects the granularity of local density estimation. A smaller K value focuses on the immediate vicinity of a feature point, resulting in a more localized density estimation, but it may also increase sensitivity to noise, as the local density measurement can be easily skewed by the presence of outlier neighbors. Conversely, a larger K value averages the distances over a broader set of neighbors, providing a more global perspective of the feature point's density in the memory bank, leading to more robust anomaly detection.

Given that the memory bank comprises patch-level features from multiple industrial images, which often contain complex information, significant intra-image distribution biases between image patches are common. Therefore, selecting a K value greater than 1 for the LDKNN algorithm can leverage these distribution biases to enhance its effectiveness. It is also observed that the optimal K value can vary across different datasets, with performance tending to stabilize or even slightly decline as K increases. Thus, identifying the optimal K value is a considerable challenge, shaped by several factors such as the size of the training set (or memory bank), the specific characteristics of the dataset, and the types of anomalies being detected. In real-world applications, employing a validation set to systematically determine the optimal K value is a practical approach.

4.4.10 Other Ablation Study on MVTec AD

To explore the impact of different hyperparameters in the REB method, this section presents additional experiments on the MVTec AD dataset using, few-shot samples,

Method \Shots	1	2	5	10	16	20	50
Im.AUROC							
SPADE	71.6	73.4	75.2	77.5	78.9	79.6	81.1
PaDim	76.1	78.9	81.0	83.2	85.5	86.5	90.1
DifferNet	-	-	-	-	87.3	-	-
Patchcore	84.1	87.2	91.0	93.8	95.5	95.9	97.7
(Ours) DefectMaker+KNN	**88.5**	**91.5**	**94.6**	**95.5**	**96.0**	**96.4**	**98.4**
(Ours) DefectMaker+LDKNN	**88.6**	**92.3**	**94.9**	**95.7**	**96.2**	**96.8**	**98.6**
Pi.AUROC							
SPADE	91.9	93.1	94.5	95.4	95.7	95.7	96.2
PaDim	88.2	90.5	92.5	93.9	94.8	95.1	96.3
Patchcore	92.4	93.3	94.8	96.1	**96.8**	**96.9**	**97.7**
(Ours) DefectMaker+KNN	**93.4**	**94.8**	**95.7**	**96.2**	96.4	96.5	96.6
(Ours) DefectMaker+LDKNN	**93.4**	**94.7**	**95.9**	**96.3**	96.5	96.6	96.8

Table 4.7: Evaluating the performance of anomaly detection with few-shot samples: Im.AUROC and Pi.AUROC metrics on MVTec AD dataset.

various backbones, and hyper-parameters of LD coefficient.

Evaluation on Few-shot Samples

This section follows the PatchCore method [57] to conduct a series of experiments evaluating few-shot learning performance (Im.AUROC and Pi.AUROC) on the MVTec AD dataset. As shown in Table 4.4.9, the proposed model is compared with four state-of-the-art anomaly detection models, using different numbers of training samples: 1, 2, 5, 10, 16, 20, and 50. Among these four baseline methods, PatchCore (ImageNet + KNN) stands out as the best performer, significantly surpassing the others. When the DefectMaker algorithm is applied to fine-tune the ImageNet-pretrained features before KNN detection (DefectMaker+KNN), an overall improvement over PatchCore across different numbers of shots was observed, demonstrating promising performance under few-shot conditions.

Evaluation on Various Backbones

The performance of REB and PatchCore with different backbones (Res18, WR50 [131], and Vgg11 [130]) on the MVTec AD dataset is shown in Fig. 4.6. Since REB already achieves excellent results on the MVTec AD dataset, larger networks than WR50 were not utilized. REB surpasses PatchCore in both anomaly detection and localization, delivering better performance even with smaller backbones, which allows for more efficient inference.

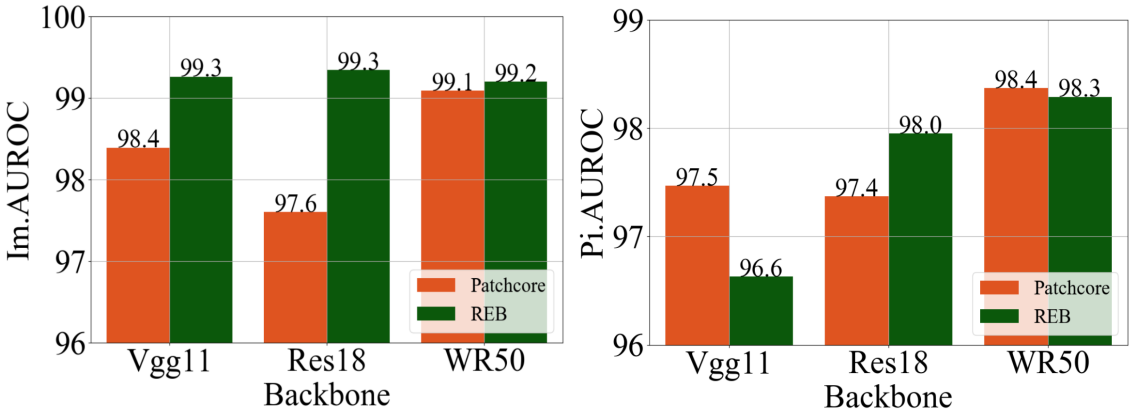

Figure 4.6: Im.AUROC (left) and Pi.AUROC (right) over different backbones on MVTec AD dataset.

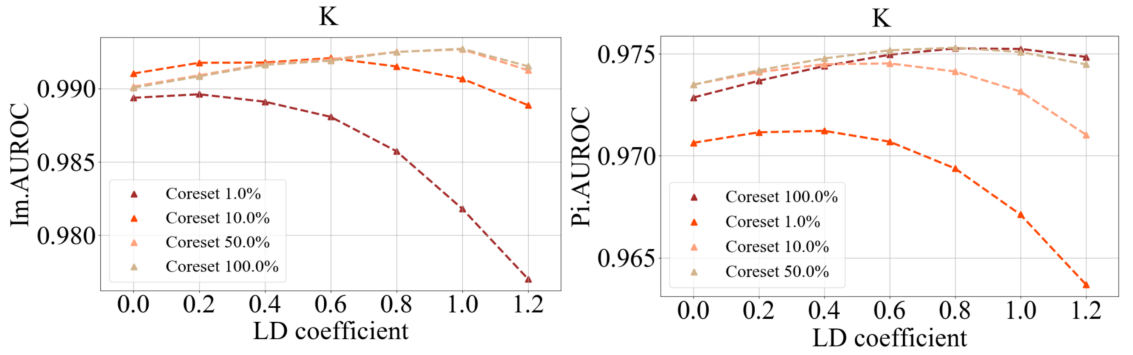

Figure 4.7: Im.AUROC (left) and Pi.AUROC (right) over different Neighbor number K and LD coefficient.

Evaluation on Various LD coefficient α

This section investigates the influence of local density bias by evaluating the changes in anomaly detection performance across different LD coefficients α in Eq. (4.3). Figure 4.7 shows the variations in Im.AUROC and Pi.AUROC with different values of α, where K in LDKNN is fixed at 9 for a fair comparison. Note that when the LD coefficient $\alpha = 0$, LDKNN reduces to 1-NN. Additionally, this section examines the relationship between local density bias and memory size by using the Coreset algorithm to down-sample the memory bank. The Coreset percentage reflects the memory bank size after down-sampling. As shown, as α increases, performance initially improves but then declines. This trend occurs because the optimal performance is achieved at different LD coefficients depending on the Coreset proportions. The larger the memory bank size, the larger the optimal α tends to be. These results

indicate that the degree of local density bias is positively related to the memory bank size. LDKNN improves anomaly detection accuracy by addressing and normalizing density bias within the feature memory bank.

4.5 Chapter Summary

This chapter has addressed the limitations of existing k-nearest neighbor (KNN) retrieval-based methods that rely on pre-trained convolutional neural network (CNN) features for industrial anomaly detection. These methods often fall short due to their limited exploitation of feature representation. To overcome these challenges, this chapter introduced the Reducing Biases (REB) approach, which targets two key biases present in the pre-trained model and feature space.The first stage of REB involves the design of the *DefectMaker* method, which generates diverse defects and employs a self-supervised learning task to mitigate domain bias in the pre-trained model, thereby enhancing the industrial-targeted feature representation. The second stage introduces the local density K-nearest neighbor (LDKNN) method, which normalizes local density bias in the patch-level feature space. This approach allows for more effective handling of complex image distributions and improves anomaly detection. We validated the effectiveness of the REB method on three real-world datasets: MVTec AD, MVTec LOCO AD, and BTAD. The method showed significant improvements in detection performance, particularly with smaller CNN models such as Vgg11 and Resnet18, achieving an impressive 99.5% Im.AUROC on the MVTec AD dataset, alongside promising inference speed. Overall, the REB method provides a practical and effective solution for industrial anomaly detection.

The classification of defects in different product surface is a crucial aspect of quality inspection in the industrial manufacturing, necessitating more extensive and thorough research. In the following chapter, the author presents an algorithm designed for the feature extraction of defect instance and the recognition of various defect categories.

Chapter 5

A General Few-shot Defect Classification Model Using Multi-View Region-Context

This chapter focuses on Few-shot defect multi-classification (FSDMC) which an emerging trend in quality control for industrial manufacturing. However, current FSDMC research often lacks generalization due to its focus on specific datasets. Additionally, defect classification heavily relies on the context within images, and existing methods fall short of effectively extracting this contextual information. To address these challenges, this chapter proposes a general FSDMC framework called MVREC, which offers two primary advantages: (1) MVREC extracts general features for defect instances by incorporating the pre-trained AlphaCLIP. (2) It utilizes a region-context framework to enhance defect features by leveraging mask region input and multi-view context augmentation. While Few-shot Zip-Adapter(-F) classifiers within the model are proposed to cache the visual features of the support set and perform few-shot classification. Furthermore, this chapter introduces MVTec-FS, a new FSDMC benchmark based on MVTec AD, which includes instance-level mask annotations and 46 defect categories. Extensive experiments conducted on MVTec-FS and fthe proposed additional datasets demonstrate the effectiveness of

MVREC.

5.1 Introduction

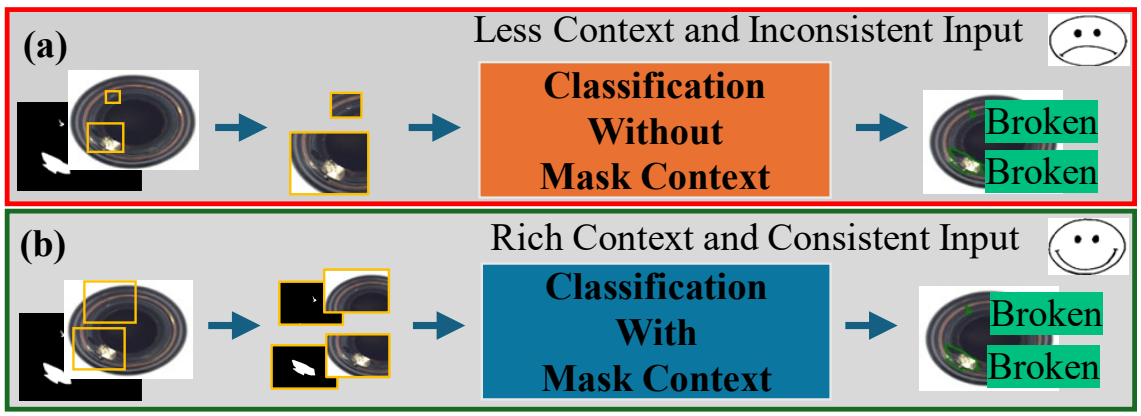

Figure 5.1: Comparison of two different Classification models.

Defect detection and classification [145] is a crucial challenge in industrial manufacturing, as it involves identifying and categorizing defects within workpieces. High-precision defect classification not only ensures the safety and reliability of products but also enhances work efficiency and reduces costs. However, in practical application scenarios, The diversity of defect types and the low occurrence of defect emergence make it a particularly difficult task.

While Few-shot Learning (FSL) has gained traction in general vision tasks like mini-Imagenet, its application to defect multi-classification (FSDMC) remains a challenge. This disparity is evident in the limited availability of dedicated datasets and research focusing on FSDMC. While Contrastive Vision-Language Pre-training (CLIP) [79] has shown remarkable success in learning visual features from large-scale image-text pairs and adapting to downstream tasks with few-shot learning, its application to FSDMC is still in its early stages. This is primarily due to the significant domain gap between general vision tasks and FSDMC. Secondly, defects inherently differ from normal surface areas, necessitating more context information for effective detection and classification. However, common classification models often involve cropping the defect region, resizing it by the model input size, and

feeding it into a network, as shown in Figure 5.1 (a). This pretreatment cannot retain the contextual information of the defect such as the surrounding background and the size of the defect. The most popular multi-category datasets [33, 34, 146] with different product images are usually for anomaly detection instead of defect classification. Although the field of few-shot defect multi-classification has attracted considerable research attention [147, 148, 149, 150, 151, 152], the datasets they used such as the NEU-DET Dataset and the MTD Dataset are limited by their focus on a single product category. There is a notable scarcity of multi-category datasets that are specifically proposed for FSDMC.

To mitigate these issues, this chapter proposes a general few-shot defect classification model using a multi-view region-context called MVREC. Specifically, the proposed approach begins with generating the region-context visual feature for the defect instance using the AlphaCLIP [89] model, a transformer-based model that takes a defect image and its mask context as input to generate the visual feature from the masked region. By incorporating the mask region context, the network can perceive the defect foreground region and its surrounding background, generating target-specific features while maintaining input consistency. Furthermore, a multi-view augmentation technique is proposed to generate multi-view features for a defect to maximize the utility of the few-shot samples and enhance generalization ability. The multi-view region-context (MVREC) features can be extracted from the multi-view patches and masks of the defect instance, thereby enhancing the region-context feature. Moreover, this chpater proposes two few-shot classifiers: the training-free Zip-Adapter-F that predicts directly without training, and the fine-tuning Zip-Adapter-F that adapts the MVREC features for better performance. Zip-Adapter and Zip-Adapter-F have the same structure, consisting of a Zero-initialized Projection (ZIP) module and a Scale-Dot-Product Attention (SDPA) module. In detail, it stores the visual features and corresponding class labels in the support set images as key-value pairs. After that, the SDPA module calculates the visual feature similarity between the query defect instance and the support defects, out-

putting the classification logits through the weighted sum of the encoded labels of the support set. The ZIP module servers as an identical mapping and feature adapter respectively for Zip-Adapter and Zip-Adapter-F.

Moreover, this chapter proposes MVTec-FS based on MVTec AD to obtain a multi-category dataset suitable for the FSDMC task. This dataset features a diverse array of categories and balanced distribution. MVTec-FS includes 15 categories of product surface images and approximately 46 types of defects, making it a promising new benchmark in this field.

This chapter tested MVREC across MVTec-FS and four other public defect datasets with classification annotations, and the results demonstrate superior performance in few-shot defect classification tasks, outperforming existing models and achieving state-of-the-art results. All in all, the contributions can be summarized as follows:

- this chapter uses AlphaCLIP and introduce a new Region-Context-based defect classification framework.
- this chapter introduces the multi-view context augmentation and Zip-adapter(-F) classifiers for few-shot classification.
- this chapter reconstructs the popular MVTec AD datasets into a new FSDMC benchmark named MVTec-FS.
- this chapter conducted rich experiments on multiple few-shot defect datasets, showing the effectiveness of MVREC.

The remainder of this chapter is structured as follows: Section 5.2 discusses the related work on defect detection and classification, few-shot learning, and the application of CLIP in defect detection tasks. Section 5.3 introduces the MVREC framework, including the multi-view region-context feature extraction and the training-free Zip-Adapter classifier. Section 5.4 presents the MVTec-FS dataset, including its construction and annotation details. Section 5.5 describes the experimental setup

and results, and Section 5.6 concludes the chapter with a summary and future research directions.

5.2 Related work

Different models, such as object detection, segmentation, and classification, have been applied to Defect Detection. These years, the MVTec AD [33] dataset has been widely researched for anomaly detection tasks [153, 113]. These anomaly detection models are designed to learn from only a group of normal samples and detect the abnormal samples. However, defect classification, identifying the defect type, is a more challenging task, as defects are rare and diverse, and related datasets are limited. Few-shot defect classification models [147, 148, 149] have been proposed. However, these methods are limited to specific datasets and require complex training processes like meta-learning and metric learning. Additionally, using part of categories as base classes to train a base model and evaluating novel categories is common but may be practical in real-world scenarios. Unlike dense prediction tasks (object detection, instance segmentation), which can predict the positions and categories of multiple instances within an image, classification networks typically provide a image-level prediction, not account for multiple instances. To address this issue, Region-context models takes position information as prompt and predicts the target's information, preserving contextual information for the target. For example, the SAM network takes prompts in the form of points, bounding boxes. To enable CLIP to focus on regions from the whole image, various methods [86, 87, 88, 89] have been explored. Alpha-CLIP is an innovative enhancement of the CLIP model, designed to augment its ability to focuse attention on specific regions by involving the introduction of an additional Alpha Conv layer parallel to the RGB Conv layer within image encoder. This architecture enables AlphaCLIP to provide precise control over the emphasis of image contents. This study uses AlphaCLIP to generate the region-context representation for the defect instance.

5.3 MVREC

This chapter introduces the MVREC visual feature extraction and then present the training-free Zip-Adapter classifier and its finetuning version Zip-Adapter-F.

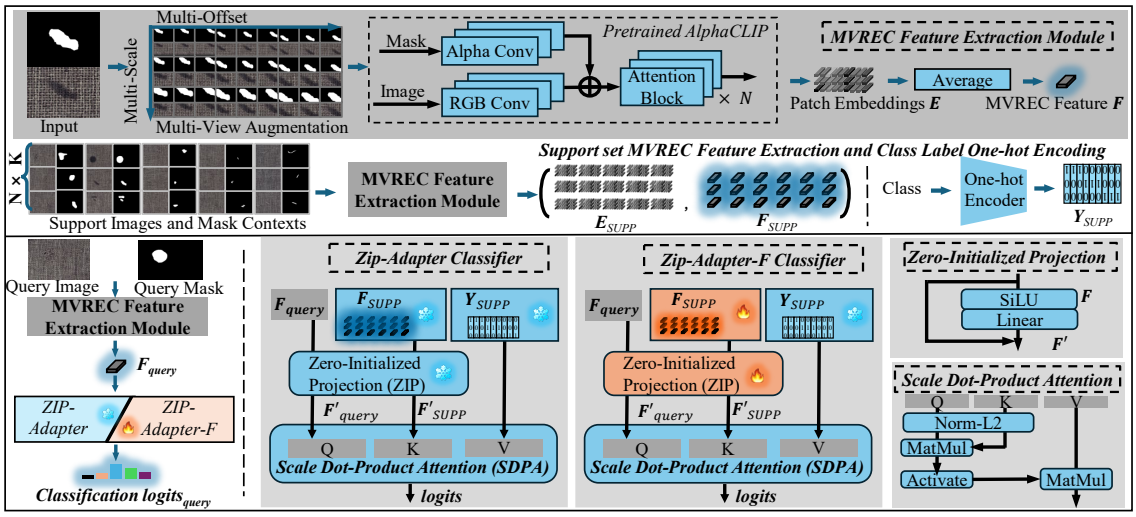

Figure 5.2: The framework of MVREC. First, the MVREC feature extraction is introduced. Given an N-way K-shot task, the MVREC features for the support set are collected. Then the inference process with Zip-Adapter or Zip-Adapter-F is shown.

5.3.1 Multi-View Region-Context Feature Extraction.

Multi-view region-context (MVREC) feature extraction for defect instances is introduced first. To effectively capture the defect visual representation and explicitly mine the context information, the pre-trained AlphaCLIP model is employed to extract visual features from images with their mask prompts. Due to the small data volume characteristic of few-shot learning tasks, this chapter employs multi-view context augmentation to generate multi-view patches of defect images to expand the available dataset for the following process. Specifically, this chapter employs two context augmentation methods to achieve this: Multi-scale augmentation is to crop Num_{scale} patches with different scales from the defect patches and their masks with the center of the defect.Subsequently, the center of the defect was offset to generate Num_{offset} defect patches with different offsets on each scale. $V = \mathrm{Num}_{scale} * \mathrm{Num}_{offset}$ patches for each defect instance can be obtained by these

two augmentation methods. The AlphaCLIP extracts MVREC patch embeddings $E \in \mathbb{R}^{V \times C}$ from the multi-view patches and obtains the MVREC feature $F \in \mathbb{R}^{C}$ by averaging E into one view.

5.3.2 Support MVREC Feature and One-Hot Label Extraction.

For N-way K-shot classification tasks, the MVREC patch embeddings $E_{SUPP} \in \mathbb{R}^{NK \times V \times C}$ and MVREC features $F_{SUPP} \in \mathbb{R}^{NK \times C}$ are extracted for the support set samples firstly. Then the one-hot-encoded class labels $Y_{SUPP} \in \mathbb{R}^{NK \times N}$ are extracted. The MVREC features F_{SUPP} and Y_{SUPP} are used to build the cached key-value pairs for the following few-shot defect classification. In addition, the MVREC patch embeddings E_{SUPP} and the Y_{SUPP} are used as training datasets to finetune the Zip-Adapter.

5.3.3 Training-free Zip-Adapter Classifier.

This section introduces the method of utilizing MVREC visual features to construct the zero-initialized projection classifier(Zip-Adapter) for few-shot defect classification tasks. The Zip-Adapter classifier consists of a zero-initialized projection (ZIP) module and a Scaled dot product attention (SDPA) Module and stores the MVREC features F_{SUPP} with encoded labels Y_{SUPP} of the support set sample. The ZIP consists of a single Linear layer, a residual connection, and a SiLU activation function, with the Linear layer initialized with zeros. The output of the ZIP module is the adapted feature F' generated with:

$$F' = \mathrm{SiLU}\left(\mathrm{Linear}\left(F\right)\right) + F \tag{5.1}$$

where F is the MVREC feature for the support sample or query sample. For Zip-Adapter, ZIP is designed to serve as an identical transformation by initializing the linear layer with zeros and residual connection. The SDPA module is a scaled

dot-product attention mechanism, which is used to calculate the visual similarity between the query defect instance and the support set and get the classification logits by the weighted sum of the support encoded labels Y_{SUPP}. The SDPA module is defined as:

$$logits_{query} = Y_{SUPP} \cdot \psi \left(Sim \left(F'_{query}, F'_{SUPP} \right) \right) \tag{5.2}$$

Where the ψ is the activation function [83] for modulating the cosine similarity:

$$\psi(x) = \exp\left(-\beta(1-x)\right), \tag{5.3}$$

β controls the sharpness of the curve. And Sim is the cosine similarity function. The output of the SDPA module $logits_{query}$ is the classification logits of the query defect instance. The class with the highest logit is identified as the predicted class.

5.3.4 Training Zip-Adapter-F classifier

Zip-Adapter-F adapts the visual features for better performance by fine-tuning the Zip-Adapter classifier, in which the ZIP module and the cached visual features of the support set are learnable. Our Zip-Adapter-F is a combination of the cache-based mechanism and the adapter-based mechanism, taking the ZIP-Adapter as the base model. The finetuning process involves two training objectives: 1) optimizing cross-entropy (CE) loss between the predicted logits $logits_{query}$ and the labels Y_{query}. The CE loss $\mathcal{L}_{CE}$ is:

$$\mathcal{L}_{CE}(logits_{query}, Y_{query}) = -\sum_{i} y_i \log(p_i) \tag{5.4}$$

where y_i and p_i represents the label and predicted probability distribution for class i.

The second part uses the triplet loss to optimize the intra-class compactness and inter-class separability of the adapted feature f_{adapted} of the ZIP module within a

batch. The triplet loss $\mathcal{L}_{triplet}$ is defined as:

$$\begin{aligned}\mathcal{L}_{triplet}(F'_{query}) = \max(d(F'_{\text{anchor}}, F'_{\text{positive}}) \\ - d(F'_{\text{anchor}}, F'_{\text{negative}}) + \alpha, 0)\end{aligned} \tag{5.5}$$

where F'_{anchor}, F'_{positive}, and F'_{negative} are the embeddings (feature vectors) of an anchor sample, a positive sample (same class as an anchor), and a negative sample (different class from anchor), within a batch, respectively. $d(\cdot, \cdot)$ is a distance function used to measure the similarity between embeddings. α is a margin hyperparameter that specifies the minimum difference between the distances of positive and negative pairs required for the loss to be zero. The overall loss function for finetuning Zip-Adapter-F is:

$$\mathcal{L}_{Zip-Adapter-F} = \mathcal{L}_{CE} + \lambda \cdot \mathcal{L}_{triplet} \tag{5.6}$$

where λ is a hyperparameter that balances the importance of the two parts in the overall loss. After Zip-Adapter-F is trained, it can be used to classify query defect instances similar to the Zip-Adapter classifier.

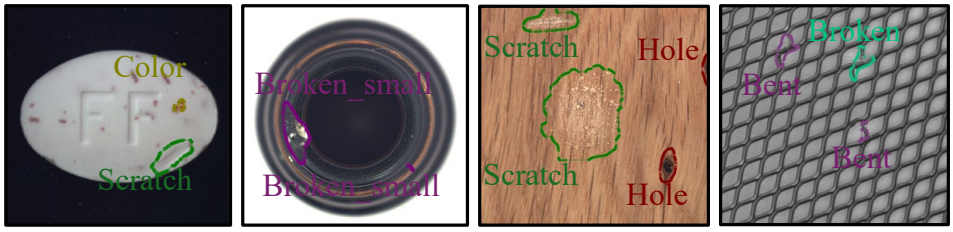

Figure 5.3: Some modified cases are displayed, in which there are multiple defect instances with different categories.

5.4 MVTec-FS Dataset

Although the field of few-shot defect multi-classification has attracted considerable research attention, the related datasets such as the NEU-DET Dataset [108] and the MTD Dataset [110] are limited by their focus on a single product category. Recently, anomaly detection research has garnered significant attention, and some multi-category datasets [33, 34, 146] with various product images have been proposed

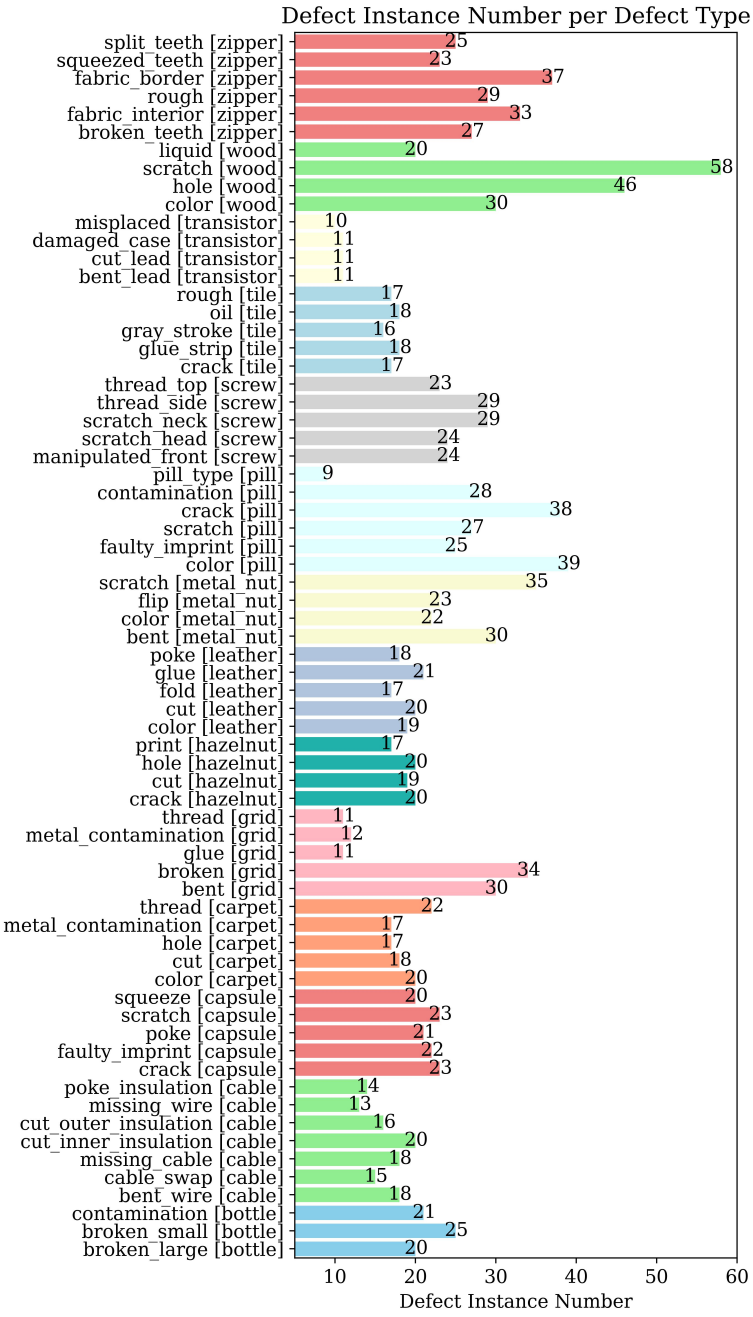

Figure 5.4: Details of MVTec-FS dataset.

for this task. However, these datasets are not proposed for defect classification. MVTec AD [33] dataset is the most popular benchmark for anomaly detection with 15 different product surface images (5 textile and 10 object), showing comprehension diversity and generalization. In the original configuration, the training set consists of lots of normal images for model training, and the testing set includes normal images and some defect images for model evaluation. The defect images are labeled with masks. There are about 47 types of defect images in total (from 8 to 26 images per category) within this dataset. This makes it suitable for few-shot defect classification tasks but Few-shot defect classification tasks are rarely studied on it even though it has classification annotations.

The 1228 defective images of MVTec AD are selected, and instance-level mask labels are labeled, making a few-shot defect multi-classification benchmark, named MVTec-FS. Since the toothbrush category contains only one defect category, this study excluded it from the proposed MVTec-FS. The number of defect instances for each defect category within each of the 46 categories (ranging from 9 to 58 defect instances per category) is presented, as shown in Fig 5.4. The original annotations were labeled at the image level and did not account for the presence of multiple defects within a single image. The connected component algorithm is used to transfer image-level masks to instance-level masks for the defect images and necessary human corrections are made. Some samples are shown in Figure 5.3. More details can be found in the appendix. In my configuration, for each defect category, 50% of the defects are used as the training set to sample the support set, and the other 50% is the testing set (query set). Few-shot setup is defined as an N-way K-shot, where K is set to 1,3, and 5. The support set is sampled from the training set, and the query set is all the testing set. Classification Accuracy is used as the evaluation metric.

FS	Classifier	Carpet	Grid	Leather	Tile	Wood	Bottle	Cable	Capsule	Hazelnut	MetalNut	Pill	Screw	Transistor	Zipper	Average
0	CLIP-ZeroShot	25.00	51.02	43.18	42.86	75.71	34.38	21.05	20.75	62.86	36.00	14.81	23.33	33.33	39.02	37.38
1	CLIP-Adapter	66.36	42.86	65.46	70.48	64.00	51.88	61.75	44.53	66.29	59.60	54.32	72.67	63.81	77.32	61.52
	CLIP-ProtoNet	60.46	42.04	62.73	70.95	63.43	50.62	62.80	46.79	62.86	62.00	59.26	74.67	55.24	74.64	60.61
	CLIP-KNN	59.55	42.04	62.73	69.53	62.86	50.62	62.80	47.17	62.86	61.60	58.52	**75.00**	55.24	74.39	60.35
	CLIP-LinearProb	60.45	41.63	64.55	73.33	68.00	55.00	62.81	49.06	69.14	60.00	63.70	73.00	56.19	73.90	62.20
	Tip-Adapter	60.00	42.45	62.73	70.00	63.43	50.62	62.80	47.17	62.86	62.00	59.26	74.67	55.24	74.64	60.56
	Tip-Adapter-F	60.91	44.90	63.18	72.86	64.57	51.25	63.16	46.79	65.14	61.20	62.71	74.34	60.95	76.59	62.04
	Zip-Adapter (MVREC)	**73.64**	**49.79**	**79.54**	**94.29**	**69.43**	**58.75**	**80.35**	**57.74**	**77.14**	**71.20**	**66.91**	68.00	**77.14**	**79.02**	**71.64**
	Zip-Adapter-F (MVREC)	**78.64**	**50.61**	**82.27**	**96.19**	**71.43**	**58.12**	**77.54**	**60.38**	**77.14**	**71.60**	**72.35**	67.33	**89.52**	**79.27**	**73.74**
3	CLIP-Adapter	70.91	57.55	77.27	89.53	81.43	68.13	77.89	61.51	76.00	72.00	68.64	85.67	88.57	85.12	75.73
	CLIP-ProtoNet	71.36	57.55	78.18	86.67	82.29	70.00	80.00	63.02	77.71	71.20	69.14	87.00	85.71	83.41	75.95
	CLIP-KNN	71.36	57.55	78.18	86.67	82.29	70.00	80.00	63.02	77.71	71.20	69.14	87.00	85.71	83.41	75.95
	CLIP-LinearProb	74.09	60.00	82.27	90.96	83.14	71.25	82.11	63.40	78.28	73.60	75.56	86.66	87.62	83.42	78.03
	Tip-Adapter	65.45	55.92	77.27	86.19	75.71	68.12	78.95	60.38	77.71	73.20	67.41	86.33	84.76	74.88	73.73
	Tip-Adapter-F	72.73	60.82	80.45	89.05	82.28	71.88	81.75	61.51	78.29	73.20	73.34	87.00	87.62	85.37	77.52
	Zip-Adapter (MVREC)	**81.82**	56.73	**87.73**	**97.14**	82.86	63.12	**91.58**	63.39	76.57	**78.40**	74.81	80.00	**97.14**	82.93	**79.59**
	Zip-Adapter-F (MVREC)	**85.00**	**71.84**	**90.91**	**97.62**	**90.00**	**76.87**	**92.98**	**73.96**	**82.29**	**83.60**	**83.46**	**88.33**	**100.00**	**88.78**	**86.12**
5	CLIP-Adapter	75.46	64.90	87.73	90.96	86.86	67.50	83.86	69.81	78.86	78.80	77.29	90.67	93.34	87.56	80.97
	CLIP-ProtoNet	73.64	59.59	83.64	89.05	84.86	67.50	84.91	72.08	74.86	78.40	74.57	89.33	**99.05**	87.07	79.90
	CLIP-KNN	74.09	55.51	81.82	89.52	79.71	65.62	78.25	62.26	73.14	79.20	67.65	89.67	97.14	79.76	76.67
	CLIP-LinearProb	79.55	66.94	88.64	93.34	88.29	70.00	86.66	69.81	78.86	80.00	81.24	**92.67**	92.38	86.34	82.48
	Tip-Adapter	72.73	59.18	80.91	88.10	84.86	63.75	82.11	65.66	77.71	78.40	67.65	90.33	98.10	78.29	77.70
	Tip-Adapter-F	74.54	65.31	88.18	90.96	89.71	68.12	85.61	70.19	77.14	80.80	80.74	91.00	96.19	87.56	81.86
	Zip-Adapter (MVREC)	**84.54**	60.00	**89.55**	**97.62**	89.43	61.88	**92.98**	**75.10**	**84.00**	**82.80**	75.06	88.00	**99.05**	83.17	**83.08**
	Zip-Adapter-F (MVREC)	**85.91**	**80.81**	**92.73**	**97.62**	**96.57**	**77.50**	**92.98**	**81.13**	**88.57**	**91.20**	**84.69**	92.00	**100.00**	**90.00**	**89.41**

Table 5.1: Classification accuracy (%) on MVTec-FS of different models with quantitative values. AlphaCLIP is used to extact the visual feature for all classifiers. The best results are highlighted in bold.

FS	MVREC	CLIP-Adapter	CLIP-ProtoNet	CLIP-KNN	CLIP-LinearProb	Tip-Adapter	Tip-Adapter-F	Zip-Adapter	Zip-Adapter-F
1	✗	61.52	60.61	60.35	62.2	60.56	62.04	60.56	**62.37**
	✓	73.05	71.59	71.29	71.82	71.64	72.95	71.64	**73.74**
	Gain	**+11.53**	+10.98	+10.94	+9.62	+11.08	+10.91	+11.08	+11.37
3	✗	75.73	75.95	70.99	**78.03**	73.73	77.52	73.73	77.93
	✓	85.68	83.4	78.91	84.76	79.59	85.72	79.59	**86.12**
	Gain	**+9.95**	+7.45	+7.92	+6.73	+5.86	+8.20	+5.86	+8.19
5	✗	80.97	79.9	76.67	**82.48**	77.7	81.86	77.7	82.24
	✓	89.21	86.14	82.59	88.11	83.08	89.22	83.08	**89.41**
	Gain	**+8.24**	+6.24	+5.92	+5.63	+5.38	+7.36	+5.38	+7.17

Table 5.2: Gains of MVREC Representation and Zip-Adapter(-F) Classifiers across different Few-Shot Settings.

5.5 Experiments

5.5.1 Experiments Setting

First, this section conducted experiments on the MVTec-FS dataset and four other datasets to evaluate the proposed MVREC using accuracy metrics. Ablation studies were performed on the MVTec-FS dataset to evaluate the effectiveness of the proposed MVREC feature extraction and the training-free Zip-Adapter classifier. For AlphaCLIP, the backbone of ViT-L14 [154] was chosen. For the MVREC visual feature, the number of scales was set to 3, representing the commonly used settings of large, medium, and small. The number of offsets was set to 9, based on a grid

Classifier	Crop style	Feature Extractor	Few Shot Setup		
			1	**3**	**5**
Zip-Adapter	Defect Size	CLIP	57.65	65.5	68.5
	Fixed Size	CLIP	58.11	64.78	68.56
	Fixed Size	AlphaCLIP Wo. Mask	61.75	68.02	70.13
	Fixed Size	AlphaCLIP	**71.64**	**79.59**	**83.08**
Zip-Adapter-F	Defect Size	CLIP	61.35	76.99	82.02
	Fixed Size	CLIP	65.09	80.57	85.47
	Fixed Size	AlphaCLIP Wo. Mask	66.32	79.59	85.44
	Fixed Size	AlphaCLIP	**73.74**	**86.12**	**89.41**

Table 5.3: Classification accuracy (%) on MVTec-FS with different region-context handling methods.

Classifier	Mulit-View Augmentation				Few Shot Setup		
	Scale	Rotate	Flip	Offset	**1**	**3**	**5**
Zip-Adapter	✗	✗	✗	✗	60.56	73.73	77.7
	✓	✗	✗	✗	68.06	78.28	81.36
	✗	✓	✗	✗	68.54	77.93	82.11
	✗	✗	✓	✗	55.27	64.78	68.16
	✗	✗	✗	✓	70.72	78.51	80.67
	✓	✓	✗	✗	70.51	78.37	81.99
	✓	✗	✓	✗	60.33	71.13	74.53
	✓	✗	✗	✓	**71.64**	**79.59**	**83.08**
Zip-Adapter-F	✗	✗	✗	✗	62.37	77.93	82.24
	✓	✗	✗	✗	69.32	82.93	86.37
	✗	✓	✗	✗	70.53	83.17	86.77
	✗	✗	✓	✗	54.69	72.62	78.32
	✗	✗	✗	✓	72.43	83.83	87.09
	✓	✓	✗	✗	73.09	84.33	88.43
	✓	✗	✓	✗	62.64	79.5	84.98
	✓	✗	✗	✓	**73.74**	**86.12**	**89.41**

Table 5.4: Classification accuracy (%) on MVTec-FS of different augmentation methods.

layout similar to a tic-tac-toe board. The β was set to 32 and 1 for the Zip-Adapter and Zip-Adapter-F classifiers, respectively. When training the Zip-Adapter-F, the AdamW optimizer with a learning rate of 0.0001 was used. The model was updated for 500 iterations, and in each iteration, the model was trained on all MVREC features of the support set. For the triplet item of the loss function, the hyper-parameters α and λ were set to 0.5 and 4, respectively.

In the experiment, a variety of baseline classifiers based on the AlphaCLIP backbone were evaluated, including:

(1) Clip-ZeroShot [79] leverages the zero-shot capability of the CLIP model. Text embeddings for each class description were generated, and the similarity between the test image embeddings and these text embeddings was computed, with classification

Zip-Adapter-F Config		Few Shot Setup		
Trainable support feature	Trainable ZIP	**1**	**3**	**5**
✓	✗	72.67	85.55	89.39
✗	✓	73.45	85.89	89.32
✓	✓	**73.74**	**86.12**	**89.41**

Table 5.5: Classification accuracy (%) on MVTec-FS of different training setttings of ZIFA-Adapter-F.

based on the highest similarity.

(2) CLIP-KNN uses the K-Nearest Neighbors algorithm with the CLIP features. The most similar K (K=1) support samples are retrieved for a query sample, and the class with the majority vote is selected as the prediction.

(3) CLIP-ProtoNet builds a Prototypical Network [155] on top of the CLIP, in which proxy features representing each class are calculated from the support set, and test images are classified based on their similarity to these class proxies.

(5) CLIP-Adapter [82] involves adding adapter layers on top of the CLIP. These layers are trained for the new classification task, adjusting the image feature feature.

(6) Tip-Adapter [83] constructs a key-value cache model via CLIP-extracted features from the few-shot data and can conduct recognition in a retrieving manner. Tip-Adapter-F treats the visual cache as learnable parameters and optimizes them to improve the performance.

5.5.2 Results on MVTec-FS Dataset

In Table 5.1, the classification accuracy (%) of various few-shot models evaluated on the MVTec-FS dataset under different few-shot learning configurations is presented. To ensure a fair comparison, this section reports results across several few-shot settings, specifically with 0, 1, 3, and 5 shots. The accuracies for 14 product categories, as well as the average accuracy, are reported in separate columns. As shown in the table, the proposed MVREC demonstrates outstanding performance in all few-shot setups, regardless of whether Zip-Adapter or Zip-Adapter-F is used. The Zip-Adapter-F achieves the highest accuracy of 89.41% with 5 shots, which is 6.93% higher than the second-best, LinearProb. By comparing different product

categories, it is evident that the Zip-Adapter-F achieves the highest accuracy in most categories, demonstrating its effectiveness in few-shot learning scenarios.

5.5.3 Ablation Study

In this section, ablation studies are conducted to investigate the contribution of different components.

Contributions of MVREC feature and Zip-Adapter(-F).

First, this section investigates the effectiveness of the MVREC feature and the Zip-Adapter(-F) classifiers by comparing the performance of the Zip-Adapter(-F) classifiers to other classifiers no matter whether the MVREC feature is used. As shown in Table 5.2, the MVREC feature consistently improves the performance of all classifiers across different few-shot settings and gets the biggest gain of 11.53% on CLIP-Adapter with 1-shot, demonstrating its general effectiveness for few-shot defect classification. Our proposed Zip-Adapter-F consistently outperforms most other classifiers, regardless of whether the MVREC feature is used. This indicates the inherent strength of the Zip-Adapter-F. When combined with the MVREC feature, the proposed Zip-Adapter-F achieves the best results across all few-shot settings. This combination maximizes the classifier's potential, making Zip-Adapter-F with MVREC the most effective approach for FSDMC. The Zip-Adapter and Zip-Adapter have the same result since they are mathematically equivalent before training.

Mask Region-Context.

This section investigates the impact mask Region-Context. As mentioned, mask Region-Context can help the model focus on the region of the defect instance without cropping the region according to the defect size which causes the loss of context information of the defect. The mask Region-Context are removed with two styles: 1) and take a whole-foreground mask as the Region-Context as the input of Alpha-clip. 2) use clip without mask Region-Context. The results from 5.3 demonstrate

that removing the mask context leads to a noticeable decrease in accuracy. This section also considers the impact of different cropping styles. Cropping by defect size with CLIP ignores the region-context. Cropping by fixed size with CLIP preserves context but doesn't use it effectively. Using AlphaCLIP Wo. (without) Mask take full foreground mask as region-context and preserves context but still underutilizes it. Finally, cropping by fixed size with AlphaCLIP and a masked region-context both preserves and effectively utilizes the context. When cropping by defect size and using vanilla CLIP, the worst results are obtained, which also indicates the importance of mask region context. Cropping by fixed size and using AlphaCLIP as the feature extractor achieves the best performance, indicating the effectiveness of MVREC.

Multi-View Context Augmentation.

From the results in Table 5.3, this section can observe that different augmentation methods have varying impacts on classification accuracy. When single augmentation is used, the multi-scale,multi-offset, and multi-rotation augmentations show significant improvements across both Zip-Adapter and Zip-Adapter-F. When double augmentations are used, the combination of multi-scale and multi-offset gets the best results, which indicates that the multi-scale and multi-offset augmentations are complementary and can be combined to achieve better performance. Multi-scale augmentation allows the model to learn features at various resolutions, which is crucial for capturing both fine and coarse details in the images. And multi-offset augmentation helps in learning robust features by shifting the image and mask context to improve the robustness and accuracy of the model in real-world applications.

Different Training Setting of Zip-Adapter(-F).

As shown in Figure 5.5, the impact of different training settings is investigated. The combination of trainable support features and trainable ZIP module leads to the highest accuracy across all few-shot setups.

5.5.4 Visualization.

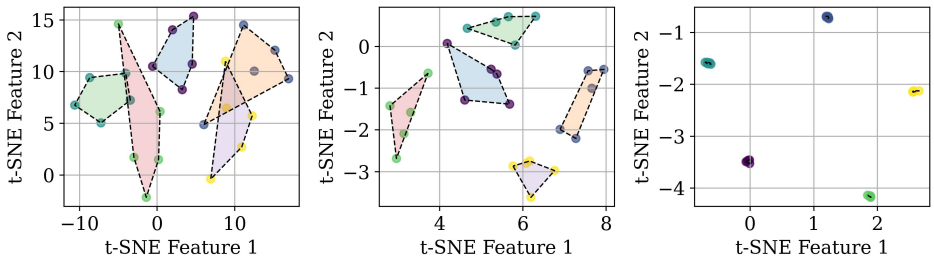

Figure 5.5: t-SNE projections of the MVREC features F_{SUPP} for support set. From left to right are 1) F_{SUPP} without multi-view augmentation. 2) F_{SUPP} 3) finetuned F_{SUPP}.

To better illustrate the function of the MVREC framework, t-SNE [156] is used to visualize the support MVREC features F_{SUPP} in Zip-Adapter-F, as shown in Figure 5.5. The dots in different colors represent 5 categories of the 5-shot leather image of the MVTec-FS dataset. The change in the distribution shows that multi-view augmentation and finetuning can help the model learn more discriminative features for classification tasks.

5.5.5 Comparison on Other Datasets

Dataset	Defect Type (Instance Number)	Annotations
NEU_DET	Crazing(689), Pitted_surface(432), Rolled_in_scale(628), Patches(881), Scratches(548), Inclusion(1011)	Bounding Bbox
PCB	Spurious_copper(503), Short(491), Spur(488), Mouse_bite(492), Missing_hole(497), Open_circuit(482)	Bounding Bbox
MTD	Break(108), Crack(69), Fray(37), Uneven(103), Blowhole(115)	Image-level Mask
AITEX	Broken_end(11), Broken_yarn(16), Broken_pick (65), Fuzzyball(42), Cuts_elvage(12), Weftcrack(15), Nep(19)	Image-level Mask

Table 5.6: Defect types and their counts in Other four datasets

Other datasets. MVREC were evaluated on several public datasets, including: **1) NEU-DET** [108] is a metal surface defect dataset for detection model research. **2) PCB Defect Dataset** [109] was released by The Open Lab on Human Robot Interaction of Peking University. **3) Magnetic Tile Surface Defects (MTD)**

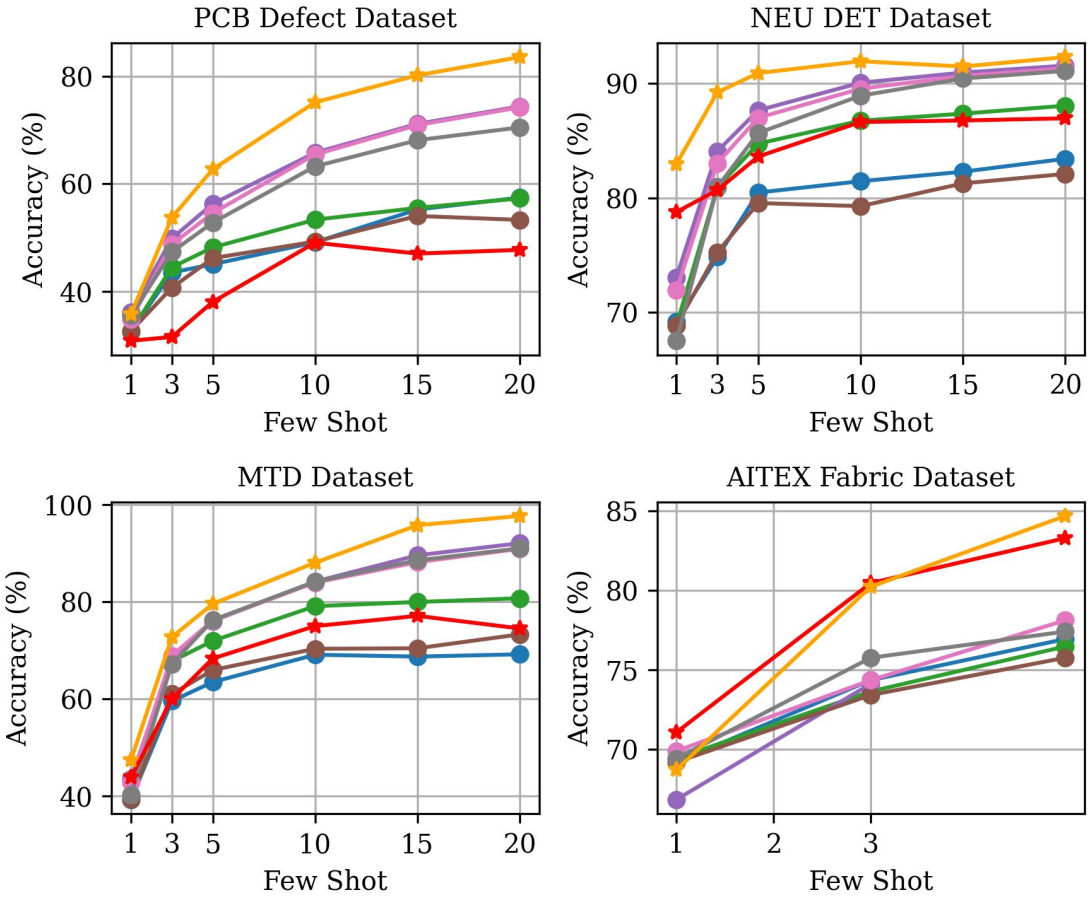

Figure 5.6: Classification accuracy (%) on other four datasets of different models with quantitative values

[110] contains 6 common magnetic tile defects. **4) AITEX Fabric defect** [111] is a fabric defect dataset with 12 types of defects, in which seven defect types with at least 10 samples are selected. For each dataset, 50% of the data are used as the training set for sampling the support set and the other 50% is the testing data (query set). In addition to 1, 3, and 5 shots, this section also evaluates the performance with 10, 15, and 20 shots on NEU-DET, PCB Defect, and MTD datasets, for a more comprehensive comparison. The results, as shown in Table 5.6 show that Zip-Adapter-F (MVREC) achieves the best performance on all datasets, and the performance of Zip-Adapter-F improves as the number of shots increases. Overall, these results demonstrate the generalization capability of the proposed method across different datasets with different types of annotations.

5.6 Chapter Summary

This chapter introduces MVREC, a general method for few-shot defect multi-classification (FSDMC). MVREC is an instance-level classification method that accommodates various labeled formats, including bounding boxes, masks, and key-points. Extensive experiments were conducted on the MVTec-FS dataset, as well as other public datasets, to evaluate the effectiveness of the proposed method, demonstrating that it is a versatile and effective approach for FSDMC tasks. Overall, MVTec-FS and MVREC can serve as benchmarks for FSDMC. I hope that this work will inspire future research in this field and contribute to the development of more advanced and effective methods. In the previous and current chapters, this chapter has addressed the tasks of defect localization and defect classification separately; however, these are often treated as independent tasks. In practical applications, end-to-end networks are generally preferred. In the next chapter, this chapter will introduce a practical end-to-end detection model using continual learning for industrial defect detection, particularly in the textile industry.

Chapter 6

A Continual Learning-based Fabric inspection Model

This chapter addresses the challenges of fabric inspection in the rapidly evolving clothing industry, where new fabric types and inspection criteria frequently emerge. Traditional inspection models struggle with pattern and inspection criterion shifts, failing to adapt to unseen fabric patterns and new defect categories. To tackle these issues, this chapter proposes a Continual Learning-based Fabric Inspection Model (CLFIM) that adapts to new inspection tasks by learning both pattern and inspection criterion contexts. This chapter pre-trained a base model on solid color fabric images and evaluated the performance of the proposed model on the MTFabric dataset with complex patterns and textures and different inspection criteria, which demonstrates that the proposed CLFIM, based on the YOLOV8 model, achieves promising results in adapting to new inspection challenges.

6.1 Introduction

In fabric inspection field, there are too many types of fabric, defects, and inspection criterion. In addition, with the clothing industry changing very fast, the production process is also constantly changing, new types of cloth may also appear, new inspection criterion may also emrge. Therefore, it is very important to design a

model for continual learning [99] on new tasks, such as different cloth types and inspection criterion. A lot of works [157, 158, 159, 160] try to solve the fabric defect inspection problem, but they do not consider the continual learning problem in the fabric inspection field. Traditional inspection models are fixed and perform poorly on patterns unseen before. The traditional inspection model is fixed and can not adapt to the new inspection criterion (classification). this chapter summarized as two problems: pattern shift and inspection criterion shift. pattern shift means that fabric of unseen pattern and texture are encountered in the new inspection task, and inspection criterion shift means that the new inspection criterion (different defect categories) is different from the previous one. These two problems are the main challenges when dealing with the unseen fabric types.

To tackle these two problems, this chapter proposes a Continual learning-based Fabric Inspection Model (CLFIM). This chapter consider this problem as a CL problem, in which the inspection model that pre-trained on a sequence of inspection tasks and adapt to new inspection task by learning pattern context and inspection criterion contxt. Pattern context and inspection criterion context are some normal images and defect samples. Learning pattern from the normal images can reduce the pattern shift from image, feature or region levels, and encourage the detectors to focus more on the defective area as a result and ignore the normal area. Learning inspection criterion can help the model to adapt to the new inspection criterion by providing a few defect samples. This study constructs two datasets: a base dataset with woven fabric images with solid color, which is used to train a common object detection model as the base model. The MTFabric dataset with complex patterns and textures and different inspection criterion, which is used to evaluate the performance of the model based YOLOV8 [161], the most popular and efficient object detection model are used as the base model to evaluate the proposed CLFIM. Rich experiments are conducted to evaluate the performance of the model, and the results show that the model can achieve promising results on the new inspection task.

6.2 Related Works

Recent advances in fabric inspection are reviewed, with a focus on methods that address pattern context and inspection criterion context. The Multistage GAN framework [157] offers a novel approach to defect inspection by addressing the complexity of varying fabric textures and defect types. By leveraging a generative adversarial network (GAN), this method is capable of synthesizing realistic defects and adapting to different fabric textures in real-world applications. The framework's integration of a deep semantic segmentation network and a multistage GAN enables continuous improvement in defect detection accuracy, making it highly relevant for tasks involving pattern context and defect inspection. The prototypical network [158] presents an advanced approach for fabric defect classification, particularly addressing challenges related to imbalanced class distributions. By utilizing a few-shot learning algorithm, this method improves classification accuracy by splitting the training set into support and query sets, ensuring a balanced representation of classes. The proposed network's ability to enhance classification in the context of few-shot fabric defects. RDDN [159] addresses the challenges of texture shift and partial visual confusion in defect detection, which are critical issues in object detection tasks within computer vision, particularly for industrial applications. The method introduces template and context references to mitigate these challenges, thereby improving the accuracy of detecting defects by focusing on the defective areas and leveraging context information for better region classification. SDANet [160] introduces a siamese defect-aware attention network designed to enhance defect detection on new, unseen samples without the need for large-scale retraining. By leveraging a siamese feature pyramid network and a defect-aware attention module, this approach highlights inconsistencies between input and template features, improving the detection accuracy for industrial applications. The method's integration as a plug-in module significantly boosts the performance of existing detection algorithms, particularly in scenarios requiring adaptive pattern context and defect detection. These methods provide valuable insights into the fabric defect detection using partten context and

inspection criterion context. However, they do not consider the continual learning problem in the fabric inspection field. This work aims to address this gap by proposing a continual learning-based fabric inspection model that can adapt to new inspection tasks by learning pattern context and inspection criterion context.

6.3 A Continual Learning-based Fabric Inspection Model

In this section, a continual learning-based fabric inspection model is proposed that evolves beyond traditional static detection approaches. The model use integreated deep defect representations extracted from a pre-trained YOLOV8 model and adapts to new inspection tasks by learning pattern context and inspection criterion context, as shown in Figure 6.2. The model is designed to address the challenges of pattern shift and inspection criterion shift in fabric inspection, enabling it to adapt to new fabric types and inspection criteria effectively.

6.3.1 YOLOV8 for Fabric Inspection

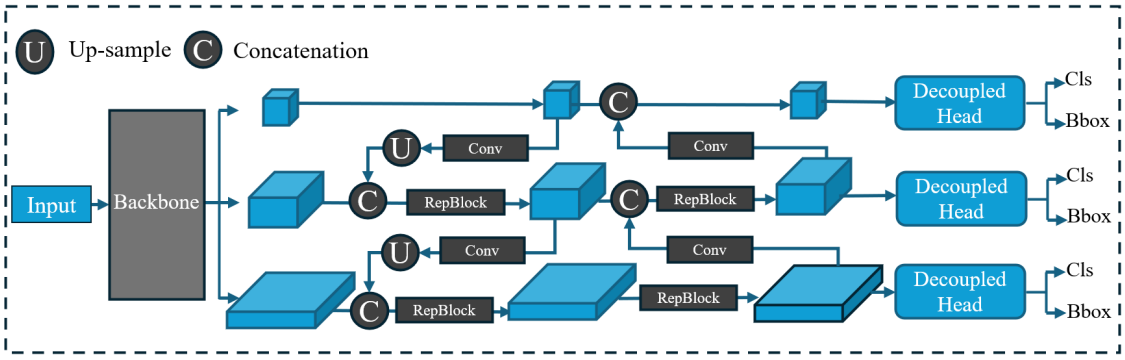

Figure 6.1: The pipeline of YOLOv8 architecture adapted for defect inspection.

Illustrated in Figure 6.1, YOLOV8 is a state-of-the-art defect detection model widely used in computer vision applications, known for its real-time detection capabilities. This backbone extracts features from the input image, which are then passed through a series of convolutional layers to predict the bounding boxes and

class labels of the detected defects in fabric images. The model employs a combination of supervised and unsupervised learning techniques to improve its performance across a wide range of defect detection tasks. Despite its relatively lower accuracy compared to two-stage detection algorithms, YOLOv8's speed and simplicity make it an ideal choice for defect inspection, where rapid and efficient processing is crucial. The network structure of YOLOv8, as applied to defect inspection, is illustrated in Figure 6.1. In defect inspection, the variability in size and shape of defects poses challenges in accurately localizing defects using traditional anchor-based methods. YOLOV8, unlike its predecessor YOLOv5, adopts an anchor-free approach. YOLOV8's Center-based method, on the other hand, identifies the central area of the defect and predicts the distances from the center to the four edges of the bounding box. This approach eliminates the dependency on predefined anchor boxes, allowing the model to learn the shape of various bounding boxes during training. Consequently, YOLOV8 enhances the efficiency and accuracy of object prediction with fewer predicted boxes, making it well-suited for detecting defects in fabrics, where defect types and sizes can vary significantly. YOLOV8 retains the binary cross-entropy (BCE) loss for classification, directly outputting confidence scores for each defect class and selecting the highest score as the final confidence. In the training process, In defect inspection, defects are typically sparse within an image dominated by non-defective background. Focusing solely on identifying defects can lead to false positives, so it is equally important to train the model on non-defective regions. However, this creates an imbalance as defect regions (positive samples) are far fewer than non-defective regions (negative samples). A dynamic assignment strategy, named Task Alignment Learning (TAL), is used to adjust sample weights dynamically during training. TAL initially penalizes misclassified samples heavily and gradually refines the model to focus on accurately detecting challenging defects. It improves the model's performance by prioritizing high-quality positive samples using combined classification scores and IoU metrics. This dynamic approach ensures more efficient learning from diverse defect data, enhancing overall model accuracy.

This section trains the YOLOV8 model on the soild color fabric dataset to serve as the foundational model for continual learning in detecting previously unseen fabric types. By leveraging YOLOV8's capabilities, this section aims to enhance the accuracy and efficiency of defect detection in fabric inspection processes.

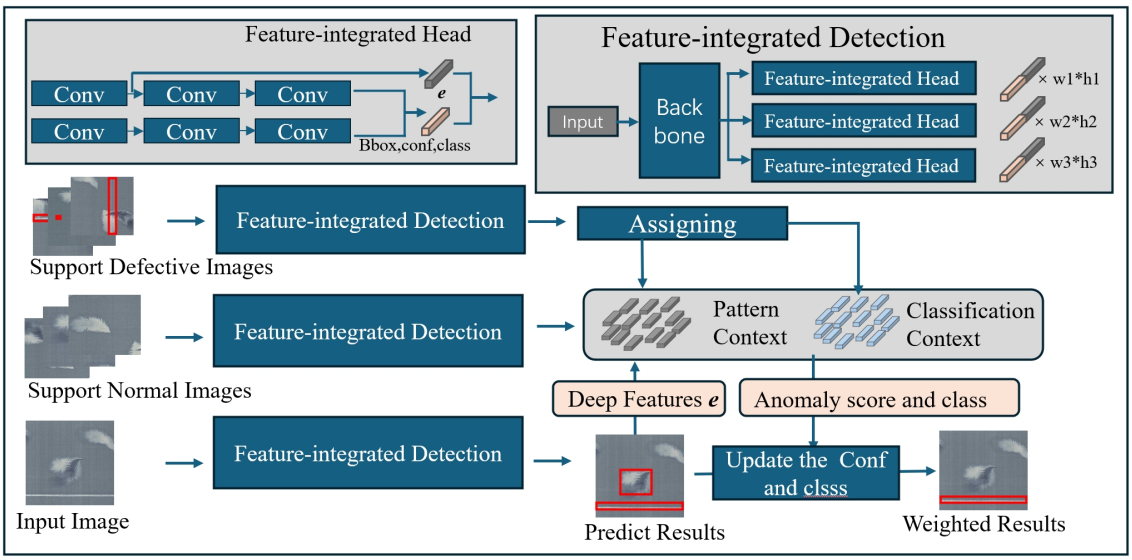

Figure 6.2: The pipeline of CLFIM.

6.3.2 Integrated Defect Representation from YOLOv8

This section focuses on extracting deep feature embeddings from the YOLOv8 model's detection head, particularly from the classification branch. The detection head of the YOLOv8 model comprises two branches: one for bounding box regression and the other for classification. The classification branch outputs class probabilities for each detected defect, while the regression branch outputs the bounding box coordinates. Once training is complete, YOLOv8's detection head can accurately identify and represent defects in fabric images. The defect representation generated by YOLOv8 includes bounding box coordinates (x_1, y_1) and (x_2, y_2), a confidence score $conf$, and a class label $class$. This representation $(x_1, y_1, x_2, y_2, conf, class)$ provides a concise summary of each detected defect, essential for automated fabric inspection systems to trigger alerts or prioritize defects based on type.

Moveover, The classification head consists of multiple convolutional layers designed to process the input feature maps and generate deep feature embeddings specific to each class. Let x_0 represent the input feature maps from the preceding layers. The classification head processes these inputs through a series of convolutional operations, mathematically expressed as:

$$e_i = \text{Conv}_1(x_{0,i}), \quad i = 1, 2, \ldots, n_l \tag{6.1}$$

where Conv_1 denotes the first convolutional layer in the classification branch, n_l is the number of detection layers, and e_i represents the feature embeddings extracted from the i-th layer. The extracted deep feature embeddings correspond to the outputs from the third-to-last layer in the classification branch. These embeddings capture high-level semantic information related to the classification task, specifically the discriminative features used to categorize defects into different classes, as learned from the training data. These deep feature embeddings are not limited to the original defect detection task for which the model was trained. They possess generalization capabilities that can be applied to other downstream tasks. this study called this module as Feature-intergrated Head, as shown in Figure 6.2. By leveraging these embeddings, this study can effectively perform task adaptation, making full use of the pre-trained knowledge encoded within the YOLOv8 model. In the context of defect inspection, the extracted deep feature embeddings, along with the bounding box (bbox) and classification results (cls), are combined to represent each detected defect, named integrated defect representation. For each detected defect, it not only provides the bounding box location and class prediction but also a high-dimensional feature vector which contain key information for classification and retain deep semantic features from the input data, ensuring that each defect has a unique deep feature representation. The advantage of this method is that it enhances the model's versatility, extending its utility beyond simple defect detection. In summary, by combining deep feature embeddings with regular detection outputs,

this approach equips each detected defect with a rich deep feature representation, significantly expanding the model's functionality and application scope.

6.3.3 Learning Pattern-Context from Normal Images

To address the pattern shift problem when encountering unseen fabrics, a pattern-context-based defect detection model designed is proposed to assess whether a detected defect corresponds to the target defect. The pattern-context is defined as a memory bank, $\mathcal{M}$, which stores integrated defect representations of normal images. The model calculates the similarity between a query sample and the memory samples to estimate the probability of the detected defect. For classifying a defect sample, the model selects the top-K most similar memory samples, based on a similarity score. The similarity score is computed using the following function:

$$\text{sim}(q, m_i) = \exp\left(-\gamma \cdot \left(1 - \frac{q \cdot m_i}{\|q\|\|m_i\|}\right)\right), \tag{6.2}$$

where γ controls the sharpness of the similarity curve, q is the query sample, and m_i represents the i-th memory sample. The similarity scores are then combined with the associated labels of the top-K memory samples to calculate the probability of the defect. This probability is given by:

$$P_{\text{prior}} = \frac{\sum_{i=1}^{K} (\text{label}_i \cdot \text{sim}(q, m_i))}{\sum_{i=1}^{K} \text{sim}(q, m_i)} \tag{6.3}$$

This probability score ranges from 0% to 100%, providing a clear indication of the confidence in the detected defect. This initial probability score is then weighted and combined with the network's direct prediction confidence to produce the final probability score:

$$\text{prob}_{\text{final}} = \alpha \cdot \text{conf}_{\text{network}} + (1 - \alpha) \cdot \text{prob}_{\text{prior}} \tag{6.4}$$

where α is a weighting factor that balances the influence of the network's direct prediction confidence ($\text{conf}_{\text{network}}$) and the calculated probability from the memory

bank ($\text{prob}_{\text{prior}}$). The final probability score determines whether the detected region is classified as a defect. This comprehensive approach effectively integrates deep feature similarity and direct network predictions, ensuring accurate and robust defect detection even in dynamic environments.

6.3.4 Learning Criterion-Context from Defect Samples

When defect samples with corresponding classification labels are available, this study can further enhance the model's adaptability by introducing a criterion-context-based detection head. This detection head adapts to new inspection criterion by learning from the a few defect samples. Unlike the pattern-context, which is derived from normal images, the criterion-context is more complex due to the introduction of new defect categories that differ from those previously encountered. The criterion-context is represented as a memory bank, $\mathcal{M}_c$, containing the prototype features [98] of each defect category. These prototype features, denoted as $\mathbf{p}_i$ for the i-th defect category, capture the central tendency of the samples within each category:

$$\mathbf{p}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{f}_j \tag{6.5}$$

where N_i is the number of samples in the i-th category, and $\mathbf{f}_j$ represents the feature embedding of the j-th sample. The model calculates the similarity between a query sample $\mathbf{q}$ and the stored prototype features $\mathbf{p}_i$ to determine the classification logits for the detected defect:

$$\text{sim}(\mathbf{q}, \mathbf{p}_i) = \exp\left(-\frac{\|\mathbf{q} - \mathbf{p}_i\|^2}{\sigma^2}\right) \tag{6.6}$$

where $\|\mathbf{q}-\mathbf{p}_i\|$ is the Euclidean distance between the query sample and the prototype feature, and σ is a scaling parameter. The defect is then classified into the category to which it is most similar, based on its proximity to the nearest prototype in the feature space:

$$\hat{y} = \arg\max_i \text{sim}(\mathbf{q}, \mathbf{p}_i) \tag{6.7}$$

This approach enables the model to incrementally learn and update its understanding of defect categories, even with a limited number of labeled samples. By leveraging this method, the model ensures robust performance in dynamic environments, continually adapting to new inspection criterion as they arise. The entire process, from feature extraction to classification, is seamlessly integrated into the detection pipeline, facilitating real-time updates and accurate defect detection.

6.4 Experiments

6.4.1 Experimental Settings

In the experiments, a standard YOLOv8 model trained on a solid color fabric dataset is used as the foundation. The solid color fabric dataset, as shown in Figure 6.3 used to train the base model is collected from the fabric factory and contains 20 defect classes, including knot, structural defect, distort, edge, tag, wrinkle, flyarn, stain, ball, attach, cusha, henji, hole, surface, ColorYarn, BrokenYarn, MissYarn, DoubleYarn, Cao, and WrongDensity. The dataset is labeled with instance-level bounding boxes. Then the model is evaluated on the MTFabric dataset with complex patterns and textures and different inspection criteria, as shown in Figure 3.3.
The model's performance is assessed across three distinct cases: (1) without utilizing pattern context or inspection criterion context, (2) utilizing pattern context only, and (3) incorporating both pattern context and inspection criterion context. This chapter focuses on binary detection performance, specifically the model's ability to distinguish between defective and non-defective samples. For Case 3, where the model must distinguish among multiple defect types, this study also assesses multi-class detection performance.

6.4.2 Evaluation Metrics

To evaluate the performance of the proposed defect detection model, this study employs three key metrics: Recall and Precision, and mean Average Precision (mAP).

Figure 6.3: Visualization of the solid color fabric dataset.

Considering that Recall and Precision vary at different threshold parameters, this section provides the Recall and Precision corresponding to the threshold at which the F1-score is maximized. Recall (R) quantifies the proportion of actual positives correctly identified by the model. It is calculated as:

$$R = \frac{TP}{TP + FN} \tag{6.8}$$

where TP represents the number of true positives, and FN denotes the number of false negatives. Precision (P) measures the proportion of predicted positives that are correctly identified. It is given by:

$$P = \frac{TP}{TP + FP} \tag{6.9}$$

where TP is the number of true positives, and FP is the number of false positives. The F1 score, the harmonic mean of Precision and Recall, provides a balanced metric

between these two measures. It is computed as:

$$F1 = 2 \times \frac{P \times R}{P + R} \tag{6.10}$$

Finally, the mean Average Precision (mAP) offers a comprehensive evaluation by averaging the Average Precision (AP) across all classes. The AP for a specific class is determined by integrating the Precision-Recall curve:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i \tag{6.11}$$

where N is the total number of classes, and AP_i denotes the Average Precision for class i. The mAP metric that this chapter uses subsequently is assumed to be mAP@50 by default. The metric mAP@50 (mean Average Precision at an Intersection over Union threshold of 50%) is commonly used to evaluate the performance of object detection models. It measures the accuracy of the model by calculating the average precision across all classes, where a detection is considered correct if the Intersection over Union (IoU) between the predicted bounding box and the ground truth box exceeds 50%.

6.4.3 Results on MTFabric Dataset

The experimental results presented in the Table 6.1 provide a clear overview of the performance improvements achieved by the proposed continual learning-based fabric inspection model across three fabric types (A, B, and C) under varying conditions: no context, pattern-context only, and a combination of pattern-context with few-shot learning (criterion-context). Across all fabric types, the baseline model, which operated without any contextual information, exhibited the lowest performance metrics, particularly in precision and recall, with Fabric Type C showing the most significant drop, where the mAP50 was just 0.219. Additionally, it is important to note that without criterion-context, the model was unable to produce multi-class classification results because the baseline model's classification criteria differed from

the test data, leading to inconsistent detection outcomes.

The introduction of pattern-context led to a marked enhancement in the model's defect detection capabilities. For example, Fabric Type C saw a significant mAP50 improvement from 0.219 to 0.641, reflecting a better generalization over standard fabric patterns. The full integration of both pattern-context and few-shot learning further bolstered the model's performance. Notably, for Fabric Type A, the mAP50 in the binary classification task increased from 0.618 to 0.635. Similarly, while the multi-class detection task showed improvements, the extent varied across fabric types.

Overall, these results underscore that the combination of pattern-context and criterion-context not only significantly improves detection performance but also enhances the model's adaptability to novel defect criteria. This combination makes the model more robust and flexible, better suited to real-world fabric inspection applications where new defect types and patterns may be encountered.

Dataset	Context		Binary			Multi-class		
	Pattern	Few-shot	Precision	Recall	mAP50	Precision	Recall	mAP50
Fabric Type A	-	-	0.536	0.587	0.577	-	-	-
	✓	-	**0.674** ↑	0.63 ↑	0.618 ↑	-	-	-
	✓	✓	0.598 ↑	**0.699** ↑	**0.635** ↑	0.284 ↑	0.543 ↑	0.401 ↑
Fabric Type B	-	-	0.682	0.314	0.393	-	-	-
	✓	-	**0.744** ↑	0.497 ↑	0.541 ↑	-	-	-
	✓	✓	0.732 ↑	**0.531** ↑	**0.554** ↑	0.29 ↑	0.375 ↑	0.327 ↑
Fabric Type C	-	-	0.289	0.589	0.219	-	-	-
	✓	-	**0.834** ↑	0.612 ↑	0.641 ↑	-	-	-
	✓	✓	0.711 ↑	**0.711** ↑	**0.699** ↑	0.395 ↑	0.52 ↑	0.502 ↑

Table 6.1: Performance comparison of the CLFIM across different fabric types and context settings.

6.4.4 Ablation Study

This section explores the influence of key hyperparameters on the performance of the proposed Continual Learning-based Fabric Inspection Model (CLFIM). The ablation study analyzes how varying these parameters impacts the model's ability to adapt to new fabric patterns and inspection criterion, addressing the challenges of pattern shift and inspection criterion shift. Ablation studies are performed on key

hyperparameters, such as α and γ, and the choice of K, which are the YOLOv8 confidence weight, the shapeness of the similarity curve, and the number of nearest neighbors, respectively.

Influence of YOLOv8 Confidence Weight α

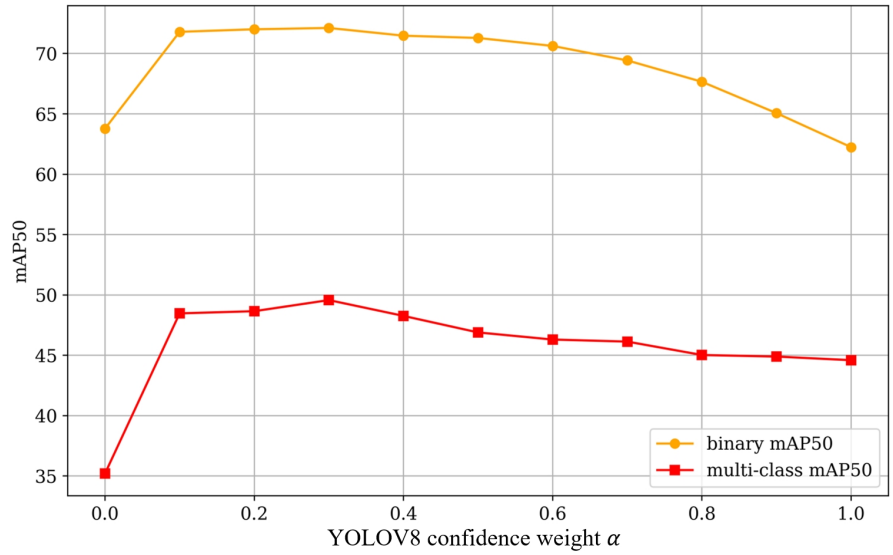

Figure 6.4: Effect of YOLOV8 confidence weight α on binary and multi-class mAP50.

The parameter α controls the weight of YOLOv8 confidence weight. As shown in the Figure 6.4, the performance trends, where both binary mAP50 and multi-class mAP50 initially increase and then decrease, demonstrate that relying solely on the YOLOv8 model's confidence score or the probability prediction based on context is not optimal. The best results are achieved when these two factors are combined with appropriate weighting. This also confirms that the introduction of pattern context significantly enhances the model's detection capability.

Influence of Nearest Neighbor Number K

The parameter K determines the number of nearest neighbors considered during the learning of pattern context. As depicted in Figure 6.5, increasing knn K leads to an upward trend in both binary mAP50 and multi-class mAP50, peaking at knn K = 50. At this point, binary mAP50 reaches the best, and multi-class mAP50

shows a similar improvement, indicating that considering a larger context of neighbors improves pattern recognition and defect identification across different classes. However, beyond knn K = 50, the performance gains plateau for both metrics, and further increases provide diminishing returns. Therefore, setting knn K around 50 is recommended for optimal defect detection performance in both binary and multi-class scenarios.

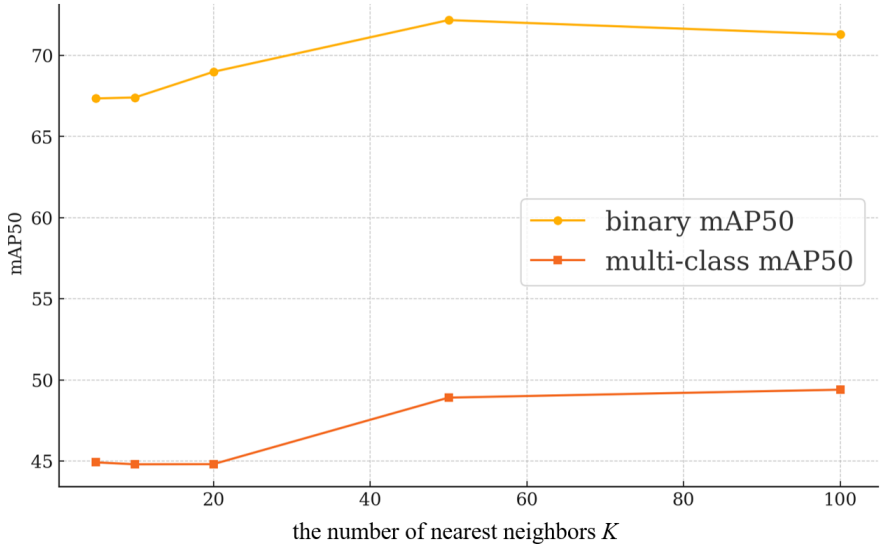

Figure 6.5: Effect of K on binary and multi-class mAP50.

Influence of the Sim Sharpness γ

The sim sharpness parameter affects the sharpness of the similarity function used during adaptation to new inspection criterion. Figure 6.6 illustrates that a setting of sim sharpness = 2.0 produces the best results. This suggests that a moderate sharpness in the similarity function allows the model to effectively discriminate between similar and dissimilar defect patterns, aiding in accurate classification of novel defect types across both binary and multi-class tasks.

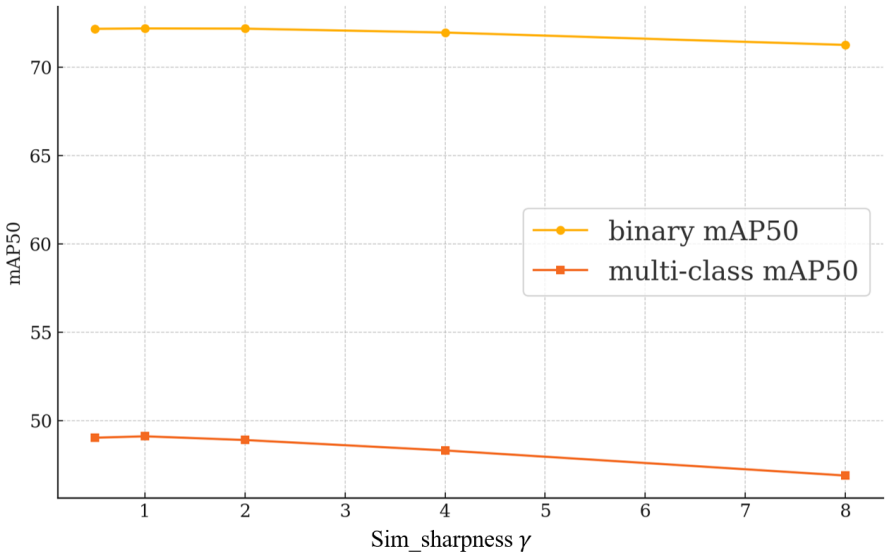

Figure 6.6: Effect of sim sharpness γ on binary and multi-class mAP50.

6.5 Visualization of the Detection Results

Figures 6.7, 6.8, and 6.9 display the detection results of the CLFIM model on three different types of datasets within the MTFabric collection. From these results, it is evident that CLFIM effectively detects defects across various datasets. Furthermore, the model successfully identifies different categories of defects specific to each dataset type. This indicates that the proposed method is adaptable and performs well under varying dataset conditions and detection criteria.

6.6 Chapter Summary

In conclusion, this chapter has developed a CL-based Fabric Inspection Model (CADDM) to address the challenges of adapting to new fabric types and inspection criterion in the rapidly evolving clothing industry. By leveraging YOLOV8 as the proposed base model, this chapter introduced the concepts of learning inspection context and inspection criterion context to mitigate the issues of pattern shift and inspection criterion shift. This approach allows the model to adapt to new inspection tasks effectively. Through comprehensive experiments on the MTFabric dataset with complex patterns and varying inspection criterion. this chapter demonstrated

Figure 6.7: Detection Results of CLFIM on Fabric Type A of the MTFabric dataset.

Figure 6.8: Detection Results of CLFIM on Fabric Type B of the MTFabric dataset.

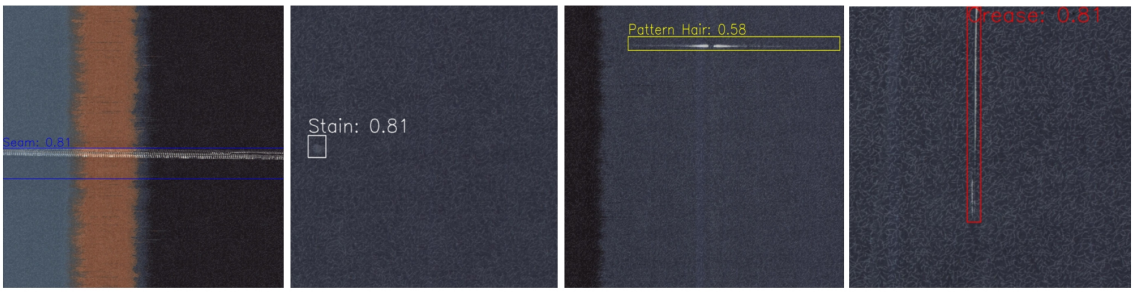

Figure 6.9: Detection Results of CLFIM on Fabric Type C of the MTFabric dataset.

that CADDM performs robustly in detecting defects under new conditions. The results highlight the model's potential in maintaining high accuracy and reliability, even as production processes and inspection criterion continue to evolve. However, this approach does have certain limitations. Currently, this model's continued learning is primarily built upon the features of the pre-trained model. While this enables the model to adapt to new tasks, further exploration is needed to optimize the pre-trained model itself for new tasks, which could lead to even better feature extraction and improved performance. Additionally, the exploration of continued learning is mainly focused on metric-based learning. Expanding this approach to include parameter optimization models is another promising direction for future research. These enhancements could further bolster the model's adaptability and accuracy in increasingly complex inspection scenarios.

Chapter 7

Conclusions and Suggestions for Future Research

7.1 Conclusions

This research has introduced several novel approaches to enhance defect detection and classification in industrial settings, particularly within computer vision-based (CV) applications. The study addresses key challenges such as the underutilization of large-scale normal samples, the difficulty of leveraging a small number of defective samples, and the need for robust detection of unseen defects. Three primary models were proposed: the Reducing Biases (REB) model for industrial anomaly detection, the MVREC framework for few-shot defect multi-classification (FSDMC), and the Continual Learning-based Fabric Inspection Model (CLFIM). Each model was developed to tackle specific issues in defect detection, such as improving feature representation, enhancing generalization capabilities, and adapting to new inspection tasks.

The REB model utilized self-supervised learning and a novel defect generation strategy, DefectMaker, to maximize the use of normal images, leading to improved anomaly detection performance. The MVREC framework integrated multi-view context augmentation and few-shot learning techniques to enhance defect classification

accuracy, particularly in scenarios with limited labeled data. Lastly, the CLFIM model, based on the YOLOV8 architecture, demonstrated robust adaptability to new fabric types and inspection criteria, showcasing its potential in rapidly evolving industrial environments.

Extensive experiments on multiple datasets, including MVTec AD, MVTec-FS, and self-collected fabric datasets, validated the effectiveness of these models. The results highlight significant improvements in detection and classification performance, offering promising solutions for real-world industrial applications.

7.2 Limitations

Despite the advancements made in this research, several limitations remain. First, while the REB model effectively reduces biases and improves anomaly detection, it demonstrated suboptimal performance on the MVTec LOCO AD dataset, particularly in detecting logical-type defects. This limitation arises from the DefectMaker strategy's inability to effectively simulate such defects. Additionally, the two phases of REB—DefectMaker and LDKNN—operate as independent modules, which may limit the overall performance.

For the MVREC framework, the study primarily focused on using image features extracted by the CLIP model's visual encoder, leaving the potential of the text encoder unexplored. This limits the framework's application in multi-modal defect detection scenarios. Moreover, the MVREC framework has not been evaluated on a unified model that can handle different defect datasets with a single training session.

The CLFIM model, while demonstrating adaptability to new inspection tasks, relies heavily on the pre-trained features of the YOLOV8 model. This dependency constrains the model's potential for further optimization in new tasks, particularly in complex and evolving industrial environments. Furthermore, the research primarily explored metric-based learning for continual learning, without delving into parameter optimization models, which could offer additional improvements in adaptability and accuracy.

7.3 Suggestions for Future Research

Future research should aim to address the limitations identified in this study. For the REB model, developing a more sophisticated DefectMaker strategy that can simulate logical defects would be a critical step toward improving its performance on datasets like MVTec LOCO AD. Additionally, integrating the two phases of REB into a cohesive learning process, potentially through a memory bank-based approach, could enhance the model's overall effectiveness.

In the case of the MVREC framework, exploring the use of CLIP's text encoder for multi-modal defect detection could expand the framework's applicability to a broader range of industrial scenarios. Furthermore, future studies should investigate the development of a unified model capable of handling various defect datasets with a single training session, which would significantly improve the framework's practicality in real-world applications.

For the CLFIM model, future research should focus on optimizing the pre-trained features for new tasks, potentially through advanced fine-tuning techniques. Expanding the exploration of continual learning to include parameter optimization models could further enhance the model's adaptability and accuracy, especially in complex inspection environments with evolving criteria.

Overall, while this research has made significant strides in improving defect detection and classification in industrial settings, ongoing efforts to address these limitations and explore new research directions will be essential to advancing the field.

References

[1] M. Pietikäinen, "Local binary patterns," *Scholarpedia*, vol. 5, no. 3, p. 9775, 2010.

[2] C. C. Gotlieb and H. E. Kreyszig, "Texture descriptors based on co-occurrence matrices," *Computer vision, graphics, and image processing*, vol. 51, no. 1, pp. 70–86, 1990.

[3] J. R. Movellan, "Tutorial on gabor filters," *Open source document*, vol. 40, pp. 1–23, 2002.

[4] C. Thibeault, "An histogram based procedure for current testing of active defects," in *International Test Conference 1999. Proceedings (IEEE Cat. No. 99CH37034)*. IEEE, 1999, pp. 714–723.

[5] R. Ren, T. Hung, and K. C. Tan, "A generic deep-learning-based approach for automated surface inspection," *IEEE transactions on cybernetics*, vol. 48, no. 3, pp. 929–940, 2017.

[6] J. Yang, S. Li, Z. Wang, H. Dong, J. Wang, and S. Tang, "Using deep learning to detect defects in manufacturing: a comprehensive survey and current challenges," *Materials*, vol. 13, no. 24, p. 5755, 2020.

[7] Z. Liu, Z. Huo, C. Li, Y. Dong, and B. Li, "Dlse-net: A robust weakly supervised network for fabric defect detection," *Displays*, vol. 68, p. 102008, 2021.

[8] J. Zhang, H. Su, W. Zou, X. Gong, Z. Zhang, and F. Shen, "Cadn: A weakly supervised learning-based category-aware object detection network for surface defect detection," *Pattern Recognition*, vol. 109, p. 107571, 2021.

[9] J. Božič, D. Tabernik, and D. Skočaj, "Mixed supervision for surface-defect detection: From weakly to fully supervised learning," *Computers in Industry*, vol. 129, p. 103459, 2021.

[10] Y.-J. Han and H.-J. Yu, "Fabric defect detection system using stacked convolutional denoising auto-encoders trained with synthetic defect data," *Applied Sciences*, vol. 10, no. 7, p. 2511, 2020.

[11] D.-M. Tsai and P.-H. Jen, "Autoencoder-based anomaly detection for surface defect inspection," *Advanced Engineering Informatics*, vol. 48, p. 101272, 2021.

[12] T. Niu, B. Li, W. Li, Y. Qiu, and S. Niu, "Positive-sample-based surface defect detection using memory-augmented adversarial autoencoders," *IEEE/ASME transactions on mechatronics*, vol. 27, no. 1, pp. 46–57, 2021.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[15] H. Alaeddine and M. Jihene, "Wide deep residual networks in networks," *Multim. Tools Appl.*, vol. 82, no. 5, pp. 7889–7899, 2023. [Online]. Available: https://doi.org/10.1007/s11042-022-13696-0

[16] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[17] N. Ma, X. Zhang, H. Zheng, and J. Sun, "Shufflenet V2: practical guidelines for efficient CNN architecture design," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11218. Springer, 2018, pp. 122–138. [Online]. Available: https://doi.org/10.1007/978-3-030-01264-9_8

[18] Q. Liang, W. Zhu, W. Sun, Z. Yu, Y. Wang, and D. Zhang, "In-line inspection solution for codes on complex backgrounds for the plastic container industry," *Measurement*, vol. 148, p. 106965, 2019.

[19] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, Conference Proceedings, pp. 1440–1448.

[20] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017. [Online]. Available: https://doi.org/10.1109/TPAMI.2016.2577031

[21] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[22] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[24] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings*

of the IEEE/CVF conference on computer vision and pattern recognition, 2023, pp. 7464–7475.

[25] X. Tao, D. Zhang, Z. Wang, X. Liu, H. Zhang, and D. Xu, "Detection of power line insulator defects using aerial images analyzed with convolutional neural networks," *IEEE transactions on systems, man, and cybernetics: systems*, vol. 50, no. 4, pp. 1486–1498, 2018.

[26] B. Hu and J. Wang, "Detection of pcb surface defects with improved faster-rcnn and feature pyramid network," *Ieee Access*, vol. 8, pp. 108 335–108 345, 2020.

[27] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[28] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 2015, pp. 234–241.

[29] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[30] ——, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[31] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[32] L. Xiao, B. Wu, and Y. Hu, "Surface defect detection using image pyramid," *IEEE Sensors Journal*, vol. 20, no. 13, pp. 7181–7188, 2020.

[33] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Mvtec ad — a comprehensive real-world dataset for unsupervised anomaly detection," pp. 9584–9592, 2019.

[34] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger, "Beyond dents and scratches: Logical constraints in unsupervised anomaly detection and localization," *Int. J. Comput. Vis.*, vol. 130, no. 4, pp. 947–969, 2022. [Online]. Available: https://doi.org/10.1007/s11263-022-01578-9

[35] W. Zhu, L. Wang, Z. Zhou, C. Wang, Y. Pan, R. Zhang, Z. Chen, L. Cheng, B.-B. Gao, J. Zhang *et al.*, "Real-iad d3: A real-world 2d/pseudo-3d/3d dataset for industrial anomaly detection," *arXiv preprint arXiv:2504.14221*, 2025.

[36] H. B. Barlow, "Unsupervised learning," *Neural computation*, vol. 1, no. 3, pp. 295–311, 1989.

[37] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[38] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.

[39] J. K. Chow, Z. Su, J. Wu, P. S. Tan, X. Mao, and Y.-H. Wang, "Anomaly detection of defects on concrete structures with the convolutional autoencoder," *Advanced Engineering Informatics*, vol. 45, p. 101105, 2020.

[40] D. Zimmerer, F. Isensee, J. Petersen, S. Kohl, and K. Maier-Hein, "Unsupervised anomaly localization using variational auto-encoders," pp. 289–297, 2019.

[41] S. Mei, Y. Wang, and G. Wen, "Automatic fabric defect detection with a multi-scale convolutional denoising autoencoder network model," *Sensors*, vol. 18, no. 4, p. 1064, 2018.

[42] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:36663713

[43] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*. Springer, 2011, pp. 52–59.

[44] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 733–742.

[45] C. Baur, B. Wiestler, S. Albarqouni, and N. Navab, "Deep autoencoding models for unsupervised anomaly segmentation in brain mr images," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 4th International Workshop, BrainLes 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers, Part I 4*. Springer, 2019, pp. 161–169.

[46] S. Mei, Y. Wang, and G. Wen, "Automatic fabric defect detection with a multi-scale convolutional denoising autoencoder network model," *Sensors*, vol. 18, no. 4, p. 1064, 2018.

[47] S. Youkachen, M. Ruchanurucks, T. Phatrapomnant, and H. Kaneko, "Defect segmentation of hot-rolled steel strip surface by using convolutional auto-encoder and conventional image processing," in *2019 10th International con-*

ference of information and communication technology for embedded systems (IC-ICTES). IEEE, 2019, pp. 1–5.

[48] J. K. Chow, Z. Su, J. Wu, P. S. Tan, X. Mao, and Y.-H. Wang, "Anomaly detection of defects on concrete structures with the convolutional autoencoder," *Advanced Engineering Informatics*, vol. 45, p. 101105, 2020.

[49] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," pp. 248–255, 2009.

[50] O. Rippel, P. Mertens, and D. Merhof, "Modeling the distribution of normal data in pre-trained deep features for anomaly detection," pp. 6726–6733, 2021.

[51] T. Defard, A. Setkov, A. Loesch, and R. Audigier, "Padim: a patch distribution modeling framework for anomaly detection and localization," pp. 475–489, 2021.

[52] Y. Zheng, X. Wang, R. Deng, T. Bao, R. Zhao, and L. Wu, "Focus your distribution: Coarse-to-fine non-contrastive learning for anomaly detection and localization," pp. 1–6, 2022.

[53] D. Gudovskiy, S. Ishizaka, and K. Kozuka, "Cflow-ad: Real-time unsupervised anomaly detection with localization via conditional normalizing flows," pp. 98–107, 2022.

[54] J. Yu, Y. Zheng, X. Wang, W. Li, Y. Wu, R. Zhao, and L. Wu, "Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows," *arXiv preprint arXiv:2111.07677*, 2021.

[55] M. Rudolph, B. Wandt, and B. Rosenhahn, "Same same but different: Semi-supervised defect detection with normalizing flows," pp. 1907–1916, 2021.

[56] M. Rudolph, T. Wehrbein, B. Rosenhahn, and B. Wandt, "Fully convolutional cross-scale-flows for image-based defect detection," pp. 1088–1097, 2022.

[57] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler, "Towards total recall in industrial anomaly detection," pp. 14 298–14 308, 2022.

[58] N. Cohen and Y. Hoshen, "Sub-image anomaly detection with deep pyramid correspondences," *arXiv preprint arXiv:2005.02357*, 2020.

[59] L. Bergman, N. Cohen, and Y. Hoshen, "Deep nearest neighbor anomaly detection," *arXiv preprint arXiv:2002.10445*, 2020.

[60] K. Sohn, C.-L. Li, J. Yoon, M. Jin, and T. Pfister, "Learning and evaluating representations for deep one-class classification," *arXiv preprint arXiv:2011.02578*, 2020.

[61] L. J. Latecki, A. Lazarevic, and D. Pokrajac, "Outlier detection with kernel density functions," vol. 7, pp. 61–75, 2007.

[62] K. L. Clarkson, "Coresets, sparse greedy approximation, and the frank-wolfe algorithm," *ACM Transactions on Algorithms (TALG)*, vol. 6, no. 4, pp. 1–30, 2010.

[63] X. Yao, Z. Chen, C. Gao, G. Zhai, and C. Zhang, "Resad: A simple framework for class generalizable anomaly detection," *Advances in Neural Information Processing Systems*, vol. 37, pp. 125 287–125 311, 2024.

[64] F. Wang, T. Zhang, Y. Wang, Y. Qiu, X. Liu, X. Guo, and Z. Cui, "Distribution prototype diffusion learning for open-set supervised anomaly detection," *arXiv preprint arXiv:2502.20981*, 2025.

[65] J. Zhu, C. Ding, Y. Tian, and G. Pang, "Anomaly heterogeneity learning for open-set supervised anomaly detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 17 616–17 626.

[66] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," *Technologies*, vol. 9, no. 1, p. 2, 2020.

[67] C.-L. Li, K. Sohn, J. Yoon, and T. Pfister, "Cutpaste: Self-supervised learning for anomaly detection and localization," pp. 9664–9674, 2021.

[68] V. Zavrtanik, M. Kristan, and D. Skočaj, "Draem-a discriminatively trained reconstruction embedding for surface anomaly detection," pp. 8330–8339, 2021.

[69] T. Tayeh, S. Aburakhia, R. Myers, and A. Shami, "Distance-based anomaly detection for industrial surfaces using triplet networks," pp. 0372–0377, 2020.

[70] Z. Li, N. Li, K. Jiang, Z. Ma, X. Wei, X. Hong, and Y. Gong, "Superpixel masking and inpainting for self-supervised anomaly detection," 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:221669098

[71] J. Tan, B. Hou, J. Batten, H. Qiu, and B. Kainz, "Detecting outliers with foreign patch interpolation," *arXiv preprint arXiv:2011.04197*, 2020.

[72] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[73] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[74] V. Zavrtanik, M. Kristan, and D. Skočaj, "Reconstruction by inpainting for visual anomaly detection," *Pattern Recognition*, vol. 112, p. 107706, 2021.

[75] X. Yan, H. Zhang, X. Xu, X. Hu, and P.-A. Heng, "Learning semantic context from normal samples for unsupervised anomaly detection," vol. 35, no. 4, pp. 3110–3118, 2021.

[76] H. M. Schlüter, J. Tan, B. Hou, and B. Kainz, "Natural synthetic anomalies for self-supervised anomaly detection and localization," pp. 474–489, 2022.

[77] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike,

and R. Lowe, "Training language models to follow instructions with human feedback," in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., 2022.

[78] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "Minigpt-4: Enhancing vision-language understanding with advanced large language models," *CoRR*, vol. abs/2304.10592, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2304.10592

[79] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[80] Z. Gu, B. Zhu, G. Zhu, Y. Chen, M. Tang, and J. Wang, "Anomalygpt: Detecting industrial anomalies using large vision-language models," in *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, M. J. Wooldridge, J. G. Dy, and S. Natarajan, Eds. AAAI Press, 2024, pp. 1932–1940. [Online]. Available: https://doi.org/10.1609/aaai.v38i3.27963

[81] J. Jeong, Y. Zou, T. Kim, D. Zhang, A. Ravichandran, and O. Dabeer, "Winclip: Zero-/few-shot anomaly classification and segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*. IEEE, 2023, pp. 19 606–19 616. [Online]. Available: https://doi.org/10.1109/CVPR52729.2023.01878

[82] P. Gao, S. Geng, R. Zhang, T. Ma, R. Fang, Y. Zhang, H. Li, and Y. Qiao, "Clip-adapter: Better vision-language models with feature adapters," *Int.*

J. Comput. Vis., vol. 132, no. 2, pp. 581–595, 2024. [Online]. Available: https://doi.org/10.1007/s11263-023-01891-x

[83] R. Zhang, W. Zhang, R. Fang, P. Gao, K. Li, J. Dai, Y. Qiao, and H. Li, "Tip-adapter: Training-free adaption of CLIP for few-shot classification," in *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXV*, ser. Lecture Notes in Computer Science, S. Avidan, G. J. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds., vol. 13695. Springer, 2022, pp. 493–510. [Online]. Available: https://doi.org/10.1007/978-3-031-19833-5_29

[84] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *Int. J. Comput. Vis.*, vol. 130, no. 9, pp. 2337–2348, 2022. [Online]. Available: https://doi.org/10.1007/s11263-022-01653-1

[85] V. Udandarao, A. Gupta, and S. Albanie, "Sus-x: Training-free name-only transfer of vision-language models," in *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*. IEEE, 2023, pp. 2725–2736. [Online]. Available: https://doi.org/10.1109/ICCV51070.2023.00257

[86] Y. Zhong, J. Yang, P. Zhang, C. Li, N. Codella, L. H. Li, L. Zhou, X. Dai, L. Yuan, Y. Li, and J. Gao, "Regionclip: Region-based language-image pretraining," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 2022, pp. 16 772–16 782. [Online]. Available: https://doi.org/10.1109/CVPR52688.2022.01629

[87] C. Zhou, C. C. Loy, and B. Dai, "Extract free dense labels from CLIP," in *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXVIII*, ser. Lecture Notes in Computer Science, S. Avidan, G. J. Brostow, M. Cissé, G. M. Farinella,

and T. Hassner, Eds., vol. 13688. Springer, 2022, pp. 696–712. [Online]. Available: https://doi.org/10.1007/978-3-031-19815-1_40

[88] A. Shtedritski, C. Rupprecht, and A. Vedaldi, "What does CLIP know about a red circle? visual prompt engineering for vlms," in *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*. IEEE, 2023, pp. 11 953–11 963. [Online]. Available: https://doi.org/10.1109/ICCV51070.2023.01101

[89] Z. Sun, Y. Fang, T. Wu, P. Zhang, Y. Zang, S. Kong, Y. Xiong, D. Lin, and J. Wang, "Alpha-clip: A clip model focusing on wherever you want," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 13 019–13 029.

[90] K. Hanbay, M. F. Talu, and Ö. F. Özgüven, "Fabric defect detection systems and methods—a systematic literature review," *Optik*, vol. 127, no. 24, pp. 11 960–11 973, 2016.

[91] K. Sakhare, A. Kulkarni, M. Kumbhakarn, and N. Kare, "Spectral and spatial domain approach for fabric defect detection and classification," in *2015 international conference on industrial instrumentation and control (ICIC)*. IEEE, 2015, pp. 640–644.

[92] J. Jing, S. Chen, and P. Li, "Automatic defect detection of patterned fabric via combining the optimal gabor filter and golden image subtraction," *Journal of Fiber Bioengineering and Informatics*, vol. 8, no. 2, pp. 229–239, 2015.

[93] Z. Wen, J. Cao, X. Liu, and S. Ying, "Fabric defects detection using adaptive wavelets," *International Journal of Clothing Science and Technology*, vol. 26, no. 3, pp. 202–211, 2014.

[94] J. Jing, D. Zhuo, H. Zhang, Y. Liang, and M. Zheng, "Fabric defect detection using the improved yolov3 model," *Journal of engineered fibers and fabrics*, vol. 15, p. 1558925020908268, 2020.

[95] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[96] A. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer, "How to train your vit? data, augmentation, and regularization in vision transformers," *Trans. Mach. Learn. Res.*, vol. 2022, 2022. [Online]. Available: https://openreview.net/forum?id=4nPswr1KcP

[97] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim, "Visual prompt tuning," in *European Conference on Computer Vision*. Springer, 2022, pp. 709–727.

[98] D. Zhou, H. Ye, D. Zhan, and Z. Liu, "Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need," *CoRR*, vol. abs/2303.07338, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2303.07338

[99] P. Janson, W. Zhang, R. Aljundi, and M. Elhoseiny, "A simple baseline that questions the use of pretrained-models in continual learning," *arXiv preprint arXiv:2210.04428*, 2022.

[100] J. Zheng, S. Qiu, and Q. Ma, "Learn or recall? revisiting incremental learning with pre-trained language models," *arXiv preprint arXiv:2312.07887*, 2023.

[101] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim, "Visual prompt tuning," in *European Conference on Computer Vision*. Springer, 2022, pp. 709–727.

[102] S. Chen, C. Ge, Z. Tong, J. Wang, Y. Song, J. Wang, and P. Luo, "Adaptformer: Adapting vision transformers for scalable visual recognition," in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans,*

LA, USA, November 28 - December 9, 2022, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., 2022.

[103] D.-W. Zhou, H.-L. Sun, H.-J. Ye, and D.-C. Zhan, "Expandable subspace ensemble for pre-trained model-based class-incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 23 554–23 564.

[104] M. D. McDonnell, D. Gong, A. Parvaneh, E. Abbasnejad, and A. van den Hengel, "Ranpac: Random projections and pre-trained models for continual learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[105] K. Ahrens, H. H. Lehmann, J. H. Lee, and S. Wermter, "Read between the layers: Leveraging intra-layer representations for rehearsal-free continual learning with pre-trained models," *arXiv preprint arXiv:2312.08888*, 2023.

[106] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger, "Beyond dents and scratches: Logical constraints in unsupervised anomaly detection and localization," *International Journal of Computer Vision*, vol. 130, no. 4, pp. 947–969, 2022.

[107] P. Mishra, R. Verk, D. Fornasier, C. Piciarelli, and G. L. Foresti, "Vt-adl: A vision transformer network for image anomaly detection and localization," pp. 01–06, 2021.

[108] Y. He, K. Song, Q. Meng, and Y. Yan, "An end-to-end steel surface defect detection approach via fusing multiple hierarchical features," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 4, pp. 1493–1504, 2020.

[109] W. Huang and P. Wei, "A pcb dataset for defects detection and classification," *arXiv preprint arXiv:1901.08204*, 2019.

[110] Y. Huang, C. Qiu, and K. Yuan, "Surface defect saliency of magnetic tile," *The Visual Computer*, vol. 36, no. 1, pp. 85–96, 2020.

[111] J. Silvestre-Blanes, T. Albero-Albero, I. Miralles, R. Pérez-Llorens, and J. Moreno, "A public fabric database for defect detection methods and results," *Autex Research Journal*, vol. 19, no. 4, pp. 363–374, 2019.

[112] "Fabric database collected from the competition of guangdong industrial intelligence innovation," http://bbs.cvmart.net/topics/706/tianchi, 2019.

[113] G. Pang, C. Shen, L. Cao, and A. van den Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 38:1–38:38, 2022. [Online]. Available: https://doi.org/10.1145/3439950

[114] X. Tao, X. Gong, X. Zhang, S. Yan, and C. Adak, "Deep learning for unsupervised anomaly localization in industrial images: A survey," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–21, 2022.

[115] S. A. Ahmed, D. P. Dogra, S. Kar, and P. P. Roy, "Trajectory-based surveillance analysis: A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 7, pp. 1985–1997, 2019.

[116] P. Seeböck, J. I. Orlando, T. Schlegl, S. M. Waldstein, H. Bogunović, S. Klimscha, G. Langs, and U. Schmidt-Erfurth, "Exploiting epistemic uncertainty of anatomy segmentation for anomaly detection in retinal oct," *IEEE Transactions on Medical Imaging*, vol. 39, no. 1, pp. 87–98, 2020.

[117] P. Seeböck, S. M. Waldstein, S. Klimscha, H. Bogunovic, T. Schlegl, B. S. Gerendas, R. Donner, U. Schmidt-Erfurth, and G. Langs, "Unsupervised identification of disease marker candidates in retinal oct imaging data," *IEEE Transactions on Medical Imaging*, vol. 38, no. 4, pp. 1037–1047, 2019.

[118] Q. Wan, L. Gao, X. Li, and L. Wen, "Industrial image anomaly localization based on gaussian clustering of pretrained feature," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 6, pp. 6182–6192, 2022.

[119] Z. Liu, Y. Zhou, Y. Xu, and Z. Wang, "Simplenet: A simple network for image anomaly detection and localization," pp. 20 402–20 411, 2023.

[120] S. Byers and A. E. Raftery, "Nearest-neighbor clutter removal for estimating features in spatial point processes," *Journal of the American Statistical Association*, vol. 93, no. 442, pp. 577–584, 1998.

[121] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," pp. 93–104, 2000.

[122] K. Zhang, M. Hutter, and H. Jin, "A new local distance-based outlier detection approach for scattered real-world data," pp. 813–822, 2009.

[123] Y. Liu, X. Gao, Z. Wen, and H. Luo, "Unsupervised image anomaly detection and localization in industry based on self-updated memory and center clustering," *IEEE Transactions on Instrumentation and Measurement*, 2023.

[124] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[125] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," *ACM Sigmod record*, vol. 28, no. 2, pp. 49–60, 1999.

[126] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Loop: local outlier probabilities," pp. 1649–1652, 2009.

[127] G. Farin, "Algorithms for rational bézier curves," *Computer-aided design*, vol. 15, no. 2, pp. 73–77, 1983.

[128] G. Li and Y. Yu, "Visual saliency detection based on multiscale deep cnn features," *IEEE transactions on image processing*, vol. 25, no. 11, pp. 5012–5024, 2016.

[129] Y.-H. Wu, Y. Liu, L. Zhang, M.-M. Cheng, and B. Ren, "Edn: Salient object detection via extremely-downsampled network," *IEEE Transactions on Image Processing*, vol. 31, pp. 3125–3136, 2022.

[130] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," pp. 770–778, 2016.

[131] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[132] J. Fan, S. Upadhye, and A. Worster, "Understanding receiver operating characteristic (roc) curves," *Canadian Journal of Emergency Medicine*, vol. 8, no. 1, pp. 19–20, 2006.

[133] S. Narkhede, "Understanding auc-roc curve," *Towards data science*, vol. 26, no. 1, pp. 220–227, 2018.

[134] T. D. Tien, A. T. Nguyen, N. H. Tran, T. D. Huy, S. Duong, C. D. T. Nguyen, and S. Q. Truong, "Revisiting reverse distillation for anomaly detection," pp. 24 511–24 520, 2023.

[135] X. Zhang, S. Li, X. Li, P. Huang, J. Shan, and T. Chen, "Destseg: Segmentation guided denoising student-teacher for anomaly detection," pp. 3914–3923, 2023.

[136] J. Hyun, S. Kim, G. Jeon, S. H. Kim, K. Bae, and B. J. Kang, "Reconpatch: Contrastive patch representation learning for industrial anomaly detection," pp. 2052–2061, 2024.

[137] H. Guo, L. Ren, J. Fu, Y. Wang, Z. Zhang, C. Lan, H. Wang, and X. Hou, "Template-guided hierarchical feature restoration for anomaly detection," pp. 6447–6458, 2023.

[138] C. Steger, "Similarity measures for occlusion, clutter, and illumination invariant object recognition," pp. 148–154, 2001.

[139] H. Park, J. Noh, and B. Ham, "Learning memory-guided normality for anomaly detection," pp. 14 372–14 381, 2020.

[140] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings," pp. 4183–4192, 2020.

[141] N. C. Tzachor, Y. Hoshen *et al.*, "Set features for fine-grained anomaly detection," *arXiv preprint arXiv:2302.12245*, 2023.

[142] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," pp. 146–157, 2017.

[143] J. Lei, X. Hu, Y. Wang, and D. Liu, "Pyramidflow: High-resolution defect contrastive localization using pyramid normalizing flow," pp. 14 143–14 152, 2023.

[144] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," *arXiv preprint arXiv:1708.00489*, 2017.

[145] S. B. Jha and R. F. Babiceanu, "Deep cnn-based visual defect detection: Survey of current literature," *Computers in Industry*, vol. 148, p. 103911, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166361523000611

[146] J. Kim, K. Jeong, H. Choi, and K. Seo, "Gan-based anomaly detection in imbalance problems," in *Computer Vision - ECCV 2020 Workshops - Glasgow, UK, August 23-28, 2020, Proceedings, Part VI*, ser. Lecture Notes in Computer Science, A. Bartoli and A. Fusiello, Eds., vol. 12540. Springer, 2020, pp. 128–145. [Online]. Available: https://doi.org/10.1007/978-3-030-65414-6_11

[147] Y. Cao, W. Zhu, J. Yang, G. Fu, D. Lin, and Y. Cao, "An effective industrial defect classification method under the few-shot setting via two-stream training," *Optics and Lasers in Engineering*, vol. 161, p. 107294, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0143816622003475

[148] Z. Zhan, J. Zhou, and B. Xu, "Fabric defect classification using prototypical network of few-shot learning algorithm," *Computers in Industry*, vol. 138, p. 103628, 2022.

[149] W. Zhao, K. Song, Y. Wang, S. Liang, and Y. Yan, "Fanet: Feature-aware network for few shot classification of strip steel surface defects," *Measurement*, vol. 208, p. 112446, 2023.

[150] C. Zhou, M. Liu, S. Zhang, P. Wei, and B. Chen, "Few-shot classification of screen defects with class-agnostic mask and context-based classifier," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–16, 2023.

[151] Z. Liu, Y. Song, R. Tang, G. Duan, and J. Tan, "Few-shot defect recognition of metal surfaces via attention-embedding and self-supervised learning," *Journal of Intelligent Manufacturing*, vol. 34, no. 8, pp. 3507–3521, 2023.

[152] W. Xiao, K. Song, J. Liu, and Y. Yan, "Graph embedding and optimal transport for few-shot classification of metal surface defect," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–10, 2022.

[153] S. Lyu, D. Mo, and W. keung Wong, "Reb: Reducing biases in representation for industrial anomaly detection," *Knowledge-Based Systems*, vol. 290, p. 111563, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705124001989

[154] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[155] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M.

Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 4077–4087.

[156] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[157] J. Liu, C. Wang, H. Su, B. Du, and D. Tao, "Multistage GAN for fabric defect detection," *IEEE Trans. Image Process.*, vol. 29, pp. 3388–3400, 2020. [Online]. Available: https://doi.org/10.1109/TIP.2019.2959741

[158] Z. Zhan, J. Zhou, and B. Xu, "Fabric defect classification using prototypical network of few-shot learning algorithm," *Comput. Ind.*, vol. 138, p. 103628, 2022. [Online]. Available: https://doi.org/10.1016/j.compind.2022.103628

[159] Z. Zeng, B. Liu, J. Fu, and H. Chao, "Reference-based defect detection network," *IEEE Trans. Image Process.*, vol. 30, pp. 6637–6647, 2021. [Online]. Available: https://doi.org/10.1109/TIP.2021.3096067

[160] Y. Zheng and L. Cui, "Defect detection on new samples with siamese defect-aware attention network," *Appl. Intell.*, vol. 53, no. 4, pp. 4563–4578, 2023. [Online]. Available: https://doi.org/10.1007/s10489-022-03595-0

[161] G. Jocher, A. Chaurasia, J. Borovec, K. Michael, T. Xie, Y. Kwon, and A. Matsushita, "YOLOv8: The latest version of the YOLO series," https://github.com/ultralytics/ultralytics, 2023, gitHub repository. [Online]. Available: https://github.com/ultralytics/ultralytics