SPGDD-GPT: Image-Text-Driven Generic Defect Diagnosis Using a Self-prompted Large Vision-Language Model

Shengwang An, and Xinghui Dong, *Member, IEEE*

***Abstract*—Large Vision-Language Models (LVLMs) mainly rely on template-generated textual descriptions to understand defects. This reliance impairs the performance of these models for Industrial Defect Detection (IDD) because they typically lack specialized knowledge. On the other hand, the majority of existing IDD methods only utilize the contrastive loss function for image-to-text feature alignment, which limits their ability to focus on defective regions. In addition, these methods usually use cosine similarity for contextual learning, which also restricts their ability to understand and adapt to complex contexts. To address these issues, we first collect a large-scale defect data set with textual descriptions, namely, the Text-Augmented Defect Data Set (TADD), to fine-tune an LVLM for defect description. We also propose a Self-prompted Generic Defect Diagnosis (including Defect Detection and Defect Description) LVLM, i.e., the SPGDD-GPT. This method can effectively utilize contextual information through a Multi-scale Self-prompted Memory Module (MSSPMM) and a Text-Driven Defect Focuser (TDDF) that we deliberately design, to adapt to unseen defect categories and focus on abnormal regions. Experimental results show that our method normally achieves the better performance than its counterparts across the 21 subsets of TADD under the 1-shot, 2-shot and 4-shot defect detection settings, demonstrating strong detection and generalization capabilities[1]. The proposed method can also generate a textural description of the defects contained in each test image. These promising results should be due to the proposed MSSPMM and TDDF and the large-scale TADD.**

***Note to Practitioners*—The proposed SPGDD-GPT is developed on top of an LVLM. It is specifically designed for the few-shot defect diagnosis task, including defect detection and defect description, which requires only a small number of training images. In real-world scenarios, the TADD effectively addresses the lack of detailed textual descriptions in training data, significantly alleviating the challenge of scarce textual data commonly encountered by practitioners in the field of defect diagnosis. By integrating a Text-Driven Defect Focuser (TDDF) and a Multi-scale Self-prompted Memory Module (MSSPMM), the SPGDD-GPT improves the alignment between visual and textual information, thereby improving the adaptability and robustness of the model in various scenarios. The TDDF explicitly adjusts the distance between normal and abnormal text embeddings through boundary hyperparameters, and achieves precise defect detection by reducing the Euclidean distance between abnormal image features and abnormal text representations, while the MSSPMM uses multi-scale normal samples as self-prompts which allow the model to rapidly adapt to novel object categories with limited samples and effectively attend to defective regions. Furthermore, the TADD consists of 35,741 images divided into 21 defect subsets with detailed textual descriptions that we annotate, providing rich contextual information. This data set facilitates the more comprehensive understanding of defect characteristics and enhances the generalizability of the model in real-world scenarios.**

***Index Terms*—Defect detection, defect diagnosis, large vision-language model, few-shot learning, vision-language alignment.**

This study was supported by the National Natural Science Foundation of China (NSFC) (No. 42576200) (Corresponding author: Xinghui Dong).

S. An and X. Dong are with the State Key Laboratory of Physical Oceanography and the Faculty of Information Science and Engineering, Ocean University of China, Qingdao, 266100.

[1]The source code, model and data set are available at https://github.com/INDTLab/SPGDD-GPT

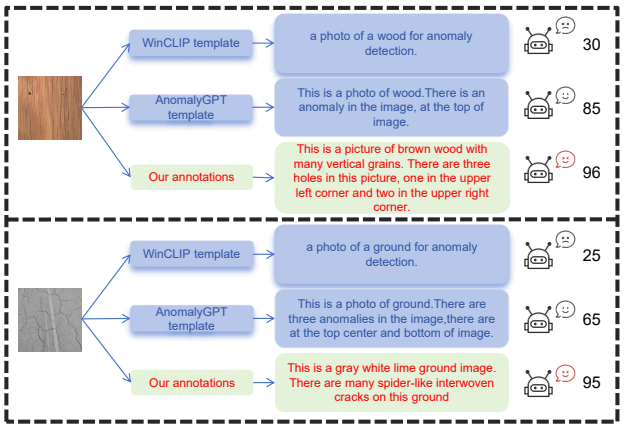


Fig. 1: Comparison between the textual descriptions obtained using two template generation methods, including WinCLIP [1] and AnomalyGPT [2], and that we manually annotated. A quality score (see Section VI-B) generated by GPT-4V [3] is shown at the far right side of each description. As can be seen, the templates that WinCLIP generated do not show an effective text prompt functionality, while the templates that AnomalyGPT generated only provide the information related to the location of defects. In contrast, our annotations not only reflect the general content of images, but also describe the location, color, shape and other characteristics of defects. These observations are consistent with the rankings of two sets of scores.

I. Introduction

INDUSTRIAL Defect Detection (IDD) is a critical step in the manufacturing process, aiming to locate and identify defects in images of industrial products. Current research on IDD is usually focused on deep learning-based methods. Although these methods [4, 5] have shown excellent performance on various IDD benchmarks, they normally require a large number of training images. In addition, the models trained often fail to perform accurate predictions in out-of-distribution or domain-shift scenarios. As a result, those methods are less feasible in practical applications.

With the advancement of Large Vision-Language Models (LVLMs), researchers have been applying them to defect

detection tasks [6, 7], aiming to enhance the semantic understanding ability of these models through prior linguistic knowledge. However, existing LVLM-based approaches still face multiple challenges in practical applications. Many methods [1, 2], which lacked textual data, could only utilize template-based text prompts, which usually failed to generate rich and diverse textual descriptions. As illustrated in Fig. 1, these descriptions failed to capture essential defect attributes, such as location, shape, color and context. This limitation impairs the semantic understanding ability of the model. Also, the feature fusion strategies that existing Vision-and-Language (V+L) segmentation methods utilized, e.g., simple multiplication [2, 8, 9, 10, 11], normally lacked the sophistication for fine-grained alignment between intricate visual characteristics of defects and detailed textual descriptions. As a result, suboptimal performance may be achieved.

Furthermore, existing few-shot IDD approaches [1, 12, 13, 14, 15, 16, 17] were normally adopted on top of the cosine similarity between normal and abnormal samples. However, this simplistic measure cannot encode rich contextual information and probably hampers effective adaptation to novel objects or defects with minimal examples. In addition, LVLMs have been pre-trained on a huge amount of ordinary Internet image-text data, which captures the insufficient domain-specific knowledge of industrial defects.

To overcome the aforementioned challenges, we introduce a novel Self-prompted Generic Defect Diagnosis LVLM, which is referred to as SPGDD-GPT. This method is able to not only perform robust and generalizable few-shot IDD tasks but also generate semantically rich textual descriptions of defects. Recent studies have revealed that the performance ceiling of LVLMs is often determined by the quality rather than the quantity of annotations. For example, Deitke et al. [18] demonstrated that a relatively small but meticulously annotated data set can outperform web-scale noisy image-text pairs in a variety of downstream tasks. Therefore, we are motivated to construct a Text-Augmented Defect Data Set (TADD), which amalgamates 21 publicly available defect data sets and contains 35,741 images in total, for the sake of solving the problem of semantically impoverished textual descriptions. Each image has been annotated with a textual description (see Fig. 1), which covers the category, location, shape, color and other characteristics of defects.

For the sake of addressing the crude feature fusion issue, we then propose a Text-Driven Defect Focuser (TDDF), which explicitly optimizes the attention of the model towards defective regions by minimizing the Euclidean distance between the image features of defects and the embeddings of corresponding abnormal textual descriptions, while maximizing the Euclidean distance between the defective images and the normal text embeddings. We further design a Multi-scale Self-prompted Memory Module (MSSPMM), to improve few-shot learning capability. This module leverages the multi-scale representation of normal samples as a self-prompt, which guides the model toward attending to defective regions and facilitates rapid adaptation to unseen defects. To bridge the domain knowledge gap without triggering catastrophic forgetting, we use the TADD to fine-tune the frozen LLM by inserting a trainable Adapter Module prior to this model. This design enables the model to effectively acquire the rich domain-specific knowledge of IDD while retaining its pre-trained general knowledge and multi-turn dialogue capability.

As a result, the proposed SPGDD-GPT can be applied to generic defect diagnosis (including defect detection and defect description) tasks. To our knowledge, the LVLM has not been explored in such a manner. Our contributions can be summarized as fourfold.

1) We collect a defect data set, that is, Text-Augmented Defect Data Set (TADD), which comprises 35,741 images divided into 21 subsets. We manually annotate a textual description for each image. To our knowledge, the TADD is the largest publicly available defect data set with detailed textual descriptions, which will benefit both the research and industrial communities.
2) We propose a Self-prompted Generic Defect Diagnosis LVLM, i.e., SPGDD-GPT, which not only performs robust and generalizable zero-shot or few-shot IDD tasks but also generates semantically rich textual descriptions of defects. In particular, a pre-trained LLM is fine-tuned using the Adapter Module. This strategy enables our model to acquire domain-specific IDD knowledge while retaining its original multi-turn dialogue capabilities and strong generalization performance.
3) We design a Text-Driven Defect Focuser (TDDF), which leverages textual descriptions to enable the model to focus on defective regions by optimizing the Euclidean distances between image features and text embeddings of normal/abnormal samples.
4) We adopt a Multi-scale Self-prompted Memory Module (MSSPMM) in order to capture contextual information from self-prompts obtained from multi-scale normal samples and use the results of zero-shot defect detection to focus on defective regions.

The rest of this paper is organized as follows. The related work is reviewed in Section II. We describe the proposed SPGDD-GPT in Section III. Our TADD is introduced in Section IV. In Sections V and VI, experimental setup and results are reported, respectively. Finally, we draw our conclusion in Section VII.

II. Related Work

A. Defect Detection

Existing defect detection methods can be categorized into traditional methods and deep learning-based methods. Traditional methods normally used hand-crafted features and a classifier to identify defects. Although these methods were interpretable and computationally efficient, they often struggled with complex or subtle defects due to their limited generalizability. Deep learning-based methods have become dominant in the past decades, which normally leverage Convolutional Neural Networks (CNNs) [19] or Vision Transformer (ViT) [20] to learn discriminative features from training images. These methods can be further divided into segmentation-based methods and classification-based methods, which are

used for pixel-level defect detection and image-level defect categorization tasks.

The above deep learning-based methods typically require a large number of annotated training images. However, it is difficult to collect such a defective data set because defects occur rarely by nature. In addition, those methods normally trained a model using a specific defect data set [21, 22]. Nevertheless, the model lacked adaptability to new scenarios. Unlike those methods, our SPGDD-GPT was built on top of recent LVLMs [2, 23]. Due to the fine-tuning of a pre-trained LLM using adapters with a relatively large comprehensive defect data set and the Text-Driven Defect Focuser (TDDF), our model can not only perform the generic defect detection task, but also generate detailed textual descriptions of defects.

B. Large Vision-Language Models

Large Vision-Language Models (LVLMs) combine the capabilities of Large Language Models (LLMs) to perform complex tasks which involve both computer vision and Natural Language Processing (NLP) applications. As a pioneering LVLM, Flamingo [24] integrated visual features into LLMs via cross-attention layers. In [25], the MoE-LLaVA model achieved efficient processing of the visual and language data through a hybrid expert system. The LLaMA 3.2 [26] model advanced multimodal research and applications by open-sourcing both its LVLM and an on-device text-only variant.

On the other hand, many existing models, such as BLIP2 [9], InstructBLIP [27] and VPGTrans [28], used a Q-Former model to feed visual features extracted using a Vision Transformer [20] into the Flan-T5 [29] model, which improved their visual knowledge acquisition capability. In particular, PandaGPT [23] enabled the multimodal input by connecting ImageBind [30] with the Vicuna [31] model using a series of linear layers.

Although the above-mentioned methods demonstrated the potential of LLM-based multimodal methods, they were normally trained on general-purpose data sets and lacked the domain-specific knowledge in IDD. In contrast, we fine-tuned the pre-trained Vicuna [31] model using annotated defective images and an adapter module. As a result, our model is able to obtain the domain-specific IDD knowledge.

C. Zero-Shot and Few-Shot Defect Detection

Zero-Shot Defect Detection (ZSDD) aims to transfer a model trained on a source domain to a target domain without using the labeled data, while Few-Shot Defect Detection (FSDD) aims to identify defects using only a limited number of normal samples in the target data set. For example, existing ZSDD models, such as WinCLIP [1], AnomalyCLIP [32], AA-CLIP [7] and AdaCLIP [6], used a pre-trained Contrastive Language-Image Pre-training (CLIP) [8] model to compare the relative distances between test images in the feature space with natural language descriptions of normal and abnormal conditions. For the sake of distinguishing abnormal instances, these distances were used to compute the scores.

FSDD models, e.g., InCTRL [14], aimed to serve as generic defect detection models by evaluating the residuals calculated between the query image and a small number of normal sample prompts. In [2], Gu et al. leveraged LVLMs to detect the presence and location of defects using a small set of normal samples for contextual learning. However, ZSDD models were often not generalizable and might struggle in real-world scenarios, in which the performance of FSDD models was inferior. Although AnomalyGPT described abnormal attributes, the descriptions were usually simplistic and might be incorrect. Nevertheless, the proposed SPGDD-GPT not only achieves high accuracy in generic defect detection tasks but also generates detailed descriptions of defects.

III. Methodology

To address the challenges that the application of LVLMs to a specific domain of IDD encounters, we propose a Self-prompted Generic Defect Diagnosis LVLM, i.e., SPGDD-GPT, which is able not only to perform robust and generalizable zero-shot or few-shot IDD tasks but also to generate semantically rich textual descriptions of defects. Fig. 2 shows the architecture of the SPGDD-GPT. As can be seen, the SPGDD-GPT comprises six components, including an image encoder adopted on top of VV-CLIP [33] (i.e., the Enhanced VV-CLIP), an original CLIP [8] text encoder, a Text-Driven Defect Focuser (TDDF), a Multi-scale Self-prompted Memory Module (MSSPMM), an Adapter Module and a pre-trained LLM (i.e., the Vicuna [31] model). The proposed method can be applied to the zero-shot and few-shot defect detection settings.

Within the zero-shot setting, image and text embeddings, extracted using the image and text encoders, respectively, are fed into the Text-Driven Defect Focuser (TDDF). The TDDF generates a series of defect-focused feature maps by modulating feature distances based on textual semantics. These maps are then used for defect detection and are converted by the Adapter Module to defect detection embeddings.

Regarding the few-shot setting, defect-focused feature maps are obtained by calculating the distance between the query image and the normal images stored in the MSSPMM and multiplying it by the output matrix of the TDDF. This distance is used as an attention weight to focus the model on defective regions. The feature maps are then converted into a format compatible with the input of the Vicuna [31] model via the Adapter Module. The Vicuna model receives the adapted output as input, which includes defect detection embeddings, query image embeddings, text embeddings and learnable embeddings. The Vicuna model further generates a comprehensive natural language response that includes a determination of whether the image contains defects, along with detailed textual descriptions specifying the location, category (e.g., scratches, dents and cracks), appearance attributes (e.g., color, shape, size and texture), and other relevant characteristics of defects observed in the image.

A. Enhanced VV-CLIP

The original VV (Vision-Vision) attention mechanism [33] can be formulated as

$$\text{Attn}(V, V, V) = \text{softmax}\left(\frac{V \cdot V^T}{\sqrt{d}}\right) \cdot V. \tag{1}$$

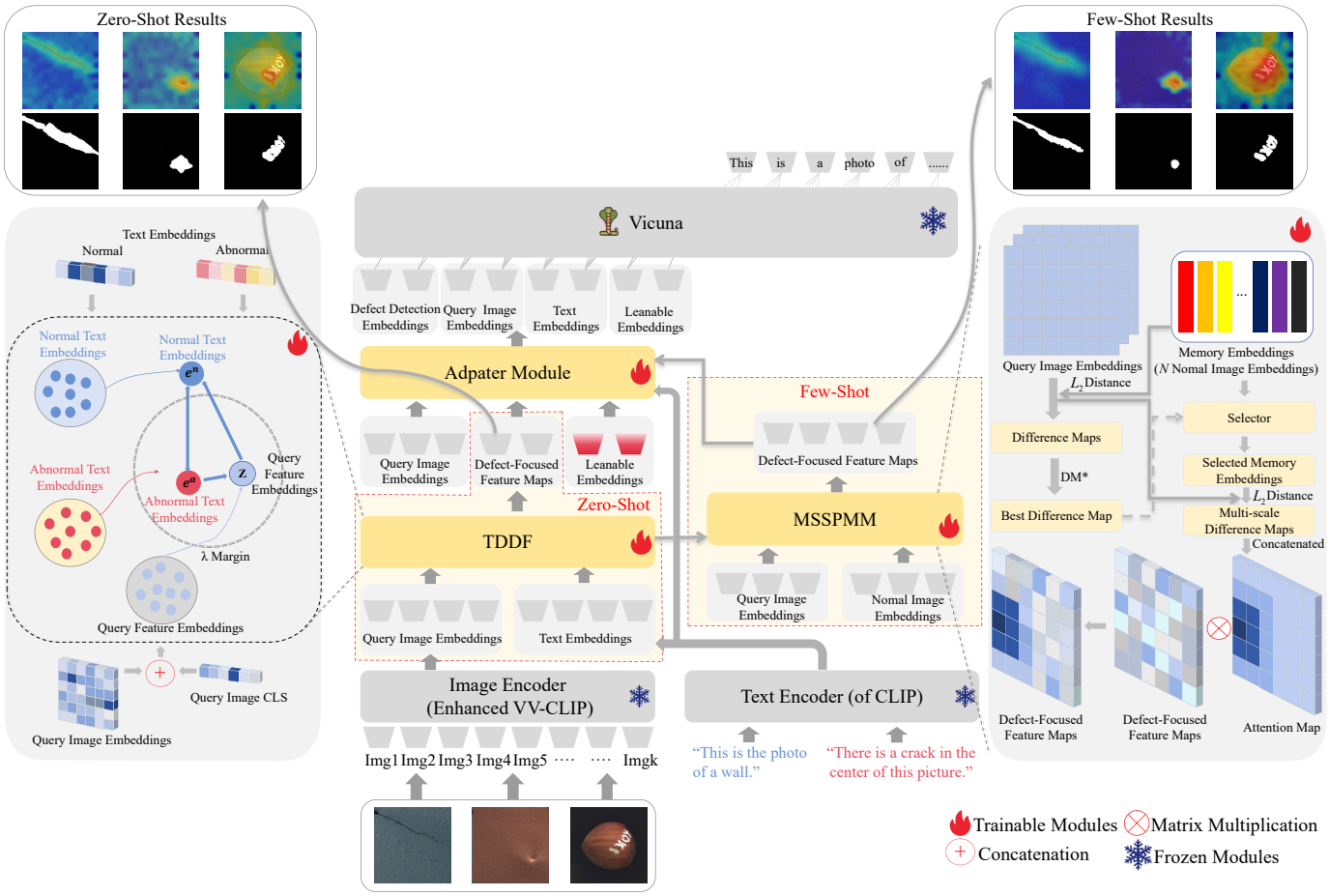


Fig. 2: The architecture of the proposed Self-prompted Generic Defect Diagnosis LVLM (SPGDD-GPT), which consists of six components, including an Enhanced VV-CLIP [33] image encoder, an original CLIP [8] text encoder, a Text-Driven Defect Focuser (TDDF), a Multi-scale Self-prompted Memory Module (MSSPMM), an Adapter Module, and a pre-trained Vicuna [31] model. Query images are first fed into the frozen Enhanced VV-CLIP [33] image encoder, to extract visual features. These features are then sent to the TDDF, which computes the distances between both normal and abnormal text embeddings and them. The result is a set of defect-focused feature maps. These feature maps along with the text embeddings, learnable embeddings and query image embeddings are transformed using the Adapter Module before being fed into the Vicuna [31] model. In particular, the result of defect detection is produced by the Adapter Module. The model further generates a detailed textual description of the defects detected. In the few-shot scenario, the MSSPMM stores the multi-scale features of normal samples and performs contextual learning by calculating the distance between query image embeddings and the most similar reference sample in the memory bank.

VV-CLIP [33] used this mechanism to obtain local features. Since global and local features were mixed together and fed into the VV attention block at each layer, local features were contaminated by global features while computational complexity was increased. This process can be formulated as

$$Z_l = \text{Proj}_l(\text{Attn}(V_l, V_l, V_l)) + Z_{l-1}^{\text{ori}}, \tag{2}$$

where Z_{l-1}^{ori} represents the global features at layer $l-1$, V_l represents the visual features at layer l, and Z_l represents the locally enhanced features at layer l.

To overcome the limitations of VV-CLIP [33] in defect detection, we revise it by building a pure local feature extraction architecture through constructing two parallel branches that completely decouple global and local operations. Our Enhanced VV-CLIP is formulated as

$$Z_l = \text{Proj}_l(\text{Attn}(V_l, V_l, V_l)). \tag{3}$$

By eliminating the global feature branch Z_{l-1}^{ori}, our method avoids the global feature contamination issue and reduces computational complexity. As a result, it can focus more on local characteristics, which are important for pixel-level detection tasks.

As shown in Fig. 3, the Enhanced VV-CLIP produces two outputs, i.e., the unaltered Z^{Ori} and the locally enhanced feature maps Z. In particular, the CLS feature $Z^{\text{Ori}}[0] \in \mathbb{R}^d$, is designated for detecting defects at the image level, whereas

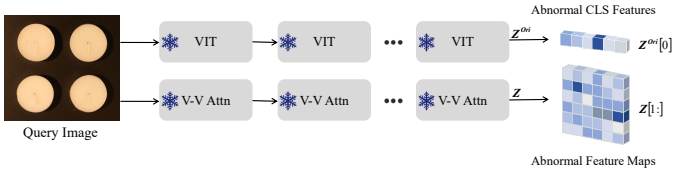


Fig. 3: The image encoder that we adopt on top of the VV-CLIP [33], i.e., the Enhanced VV-CLIP. In particular, we introduce a parallel VV attention pathway within the image encoder of CLIP [8], designed to preserve local features. This improvement is useful for the pixel-level defect detection task.

the local feature maps $Z[1:] \in \mathbb{R}^{T\times d}$, can be used for the pixel-level defect detection task.

B. Text-Driven Defect Focuser

To perform the defect detection task, existing image-text alignment methods typically concatenate normal and abnormal text embeddings and use them together with image features to compute the contrastive loss. However, these methods cannot effectively control the boundary between normal and abnormal text embeddings and cannot optimize the distance between defective image embeddings and text embeddings. To address this problem, we propose a Text-Driven Defect Focuser (TDDF), which introduces a boundary hyperparameter in order to explicitly adjust the distance between normal and abnormal text embeddings. As a result, the clearer boundary guidance can be obtained, which enables the more accurate alignment between defective image features and the corresponding abnormal text embeddings.

The TDDF aims to increase the focus on defects in the query image by minimizing the Euclidean distance between the query image embedding and the semantically rich abnormal text embeddings (which encode the descriptions of defect categories and attributes). Meanwhile, the TDDF also aims to suppress the focus on the normal regions of the query image by maximizing the distance between the query image embedding and the normal text embeddings. In this context, the TDDF can sharpen the focus on defective regions and thus boost the performance of defect detection.

Specifically, the loss function of the TDDF can be defined as

$$\mathcal{L}_{\text{TDDF}} = \max\left(0, d\left(\frac{\mathbf{z}}{\|\mathbf{z}\|^2}, \frac{\mathbf{e}^n}{\|\mathbf{e}^n\|^2}\right) - d\left(\frac{\mathbf{z}}{\|\mathbf{z}\|^2}, \frac{\mathbf{e}^a}{\|\mathbf{e}^a\|^2}\right)\right), \tag{4}$$

where $d(\cdot,\cdot)$ represents the Euclidean distance, $\mathbf{e}^n$ denotes the normal text embedding, $\mathbf{e}^a$ represents the abnormal text embedding, and $\mathbf{z}$ stands for the query image embedding. Within CLIP [8], the final features are projected onto the unit hyper-sphere. Therefore, the features in $\mathcal{L}_{\text{TDDF}}$ are normalized and the margin is set to zero constantly. Compared with the contrastive loss, i.e., $\mathcal{L}_{\text{CLIP}}$, $\mathcal{L}_{\text{TDDF}}$ ensures that the distance between normal text embeddings and abnormal image embeddings is greater than the distance between abnormal text embeddings and abnormal image embeddings.

Furthermore, we use the squared $\mathcal{L}_2$ norm to separate the distances between text embeddings

$$\mathcal{L}_{\text{Sep}} = \lambda \cdot \left\| \frac{e^n}{\|e^n\|^2} - \frac{e^a}{\|e^a\|^2} \right\|_2^2, \tag{5}$$

where λ is a hyperparameter used to control the degree of separation of text features. Due to the use of $\mathcal{L}_{\text{Sep}}$, we can not only control the distance between normal and abnormal text embeddings, but also optimize the boundary between the defective area and the normal area of the query image.

The Euclidean distance d_a between the query image embedding and the abnormal text embedding and the Euclidean distance d_n between the query image embedding and the normal text embedding are computed. A set of defect-focused feature maps can be generated by calculating the difference between the two distance values:

$$\Delta d = d_n - d_a. \tag{6}$$

As a result, the defect region is enhanced. Δd is then jointly optimized with the loss function $\mathcal{L}_{\text{TDDF}}$, to ensure that the defect-focused feature maps produce high response values in abnormal regions and maintain low response values in normal regions. Clear segmentation of the edge of defects can be achieved through the gradient constraint of $\mathcal{L}_{\text{Sep}}$. This dual-loss design ensures that defective regions are precisely localized while maintaining clear boundaries between normal and defective regions.

C. Multi-scale Self-prompted Memory Module

Existing few-shot learning methods [1, 32, 33] mainly relied on cosine similarity for contextual learning. They usually failed to capture subtle local variations in defective images. To address this limitation, we propose a Multi-scale Self-prompted Memory Module (MSSPMM) that establishes hierarchical memory embeddings from normal samples and leverages them as self-prompts to enrich contextual information learning. This module integrates multi-scale feature analysis with dynamic prompt selection in order to enhance defect localization.

To be specific, we randomly select N normal images as memory samples. Multi-scale embeddings are extracted from them using blocks 3, 8 and 12 of the Enhanced VV-CLIP image encoder. These embeddings capture complementary visual characteristics in different receptive fields. This hierarchical representation is formalized as

$$\mathbf{ME} = \{\mathbf{ME}^{(3)}, \mathbf{ME}^{(8)}, \mathbf{ME}^{(12)}\}, \tag{7}$$

where block 3 focuses on local characteristics through embeddings of dimension $N \times 225 \times 896$, block 8 captures mid-level semantics with dimension $N \times 225 \times 768$, and block 12 encodes global context in dimension $N \times 225 \times 640$. Given a query image, we similarly extract multi-scale embeddings

$$\mathbf{QE} = \{\mathbf{QE}^{(3)}, \mathbf{QE}^{(8)}, \mathbf{QE}^{(12)}\}, \tag{8}$$

and compute layer-wise L_2 distance maps between the query image embeddings and the memory embeddings

$$\mathbf{DM} = \bigcup_{i=1}^{N} \sum_{s\in\{3,8,12\}} \|\mathbf{QE}^{(s)} - \mathbf{ME}_i^{(s)}\|_2. \tag{9}$$

The optimal difference map in terms of each memory sample is further selected according to:

$$\mathbf{DM}_i^* = \arg\min_{\mathbf{DM}_i} \sum_{x \in \mathbf{DM}_i} x, \tag{10}$$

which enables dynamic selection of the most relevant multi-scale memory embeddings as self-prompts. After determining the best difference map among the optimal difference maps across the memory samples, we select the multi-scale memory embedding corresponding to the best difference map as the self-prompt and calculate multi-scale difference maps with the query image using this prompt. Finally, we concatenate these multi-scale difference maps along the channel dimension to generate a defect attention map, $\mathbf{AM} \in \mathbb{R}^{H \times W \times C}$.

This map is multiplied element-wise by the defect-focused feature maps ($\mathbf{DFFM}$) to learn contextual information. This process can be expressed as:

$$\mathbf{F}^{Out} = \mathbf{AM} \odot \mathbf{DFFM}, \tag{11}$$

where $\odot$ denotes the Hadamard product and $\mathbf{F}^{Out}$ represents the resultant defect-focused feature maps. $\mathbf{F}^{Out}$ is then fed into the Adapter Module for adaptation transformation. As a result, defect detection embeddings are produced, which can be used as the input of the Vicuna [31] model. In essence, the MSSPMM is able to effectively enhance contextual reasoning under the few-shot settings by simultaneously preserving local discriminability through fine-grained block 3 features and maintaining global consistency via high-level block 12 representations. In addition, block 8 features provide intermediate semantic bridging.

D. Adapter Module

Inspired by studies on Prompt-Tuning in Parameter-Efficient Fine-Tuning (PEFT) [34], we freeze the backbone and prepend a lightweight Adapter Module that injects downstream defect knowledge through prompt-tuning. Specifically, a small set of learnable embeddings, trained on the manually annotated TADD, are concatenated to the multi-modal input. Since prompt-tuning uses far smaller number of parameters than LoRA [35], the module is less prone to be overfitted in low-data regimes and more robust when there is a large domain gap. The Adapter Module consists of multiple linear layers, a small set of learnable embeddings, and a concatenation operation. Each linear layer receives input embeddings of different dimensions, but the output dimensions remain consistent. The concatenation operation combines the outputs of those linear layers, including defect detection embeddings, original query image embeddings and text embeddings, with the learnable embeddings and feeds them into the Vicuna [31] model. Finally, the output of the model is a textual description of the detected defects.

E. Loss Functions

To train our SPGDD-GPT, we employ four loss functions, including $\mathcal{L}_{\text{C}}$ (Cross-Entropy Loss), $\mathcal{L}_{\text{I}}$ (Cross-Entropy Loss), $\mathcal{L}_{\text{Sep}}$ and $\mathcal{L}_{\text{TDDF}}$. The cross-entropy loss function $\mathcal{L}_{\text{c}}$ is used to supervise the generation of textual descriptions of defects by the Vicuna [31] model. This function measures the discrepancy between the tokens that the Vicuna model generates and the ground-truth textual description contained in the TADD in terms of an image. The function can be formulated as:

$$\mathcal{L}_C = -\sum_{i=1}^{n} y_i \log(p_i), \tag{12}$$

where n is the number of tokens, y_i is the ground-truth label for token i and p_i is the predicted probability for token i.

To supervise the results of defect detection that the Adapter Module generates, the cross-entropy loss function $\mathcal{L}_{\text{I}}$ is employed. This function measures the difference between the results of defect detection and the ground-truth of the corresponding defective images. $\mathcal{L}_{\text{I}}$ can be expressed as:

$$\mathcal{L}_I = -\sum_{p=1}^{n} y_p \log(p_p), \tag{13}$$

where n denotes the number of pixels, y_p stands for the ground-truth label of pixel i in an image and p_i represents the predicted probability that the model produces at pixel i.

Finally, the overall loss function is defined as:

$$\mathcal{L} = \alpha\mathcal{L}_C + \beta(\mathcal{L}_{\text{TDDF}} + \lambda\mathcal{L}_{\text{Seq}}) + \gamma\mathcal{L}_I, \tag{14}$$

where α, β and γ are the coefficients used to balance the four loss functions. These coefficients were set to 1 by default in our experiments.

IV. Text-Augmented Defect Data Set

Although there are many publicly available defect data sets, they normally lack textual descriptions related to defects. This issue prevents LLMs from being applied to these data sets. To overcome this challenge, we collected 21 publicly available defect data sets, which contained 35,741 images in total, and manually annotated each image using a textual description. These images and the associated textual descriptions are comprised of a data set, referred to as the Text-Augmented Defect Data Set (TADD).

The 35,741 images were captured from various samples, such as fabrics, pavement, walls, electron commutators, tiles, steel strips, tires, etc., under different conditions. The diversity of these images enables the network to learn rich characteristics of various defects related to different samples. As a result, the generalizability of the model trained can be improved for different real-world applications.

A. Publicly Available Data Sets

In this subsection, we briefly introduce the 21 publicly available data sets. For more details, please refer to the original publications.

1) Aitex: The Aitex [36] data set contains 245 fabric images of seven textile structures. In total, 12 categories of defects are included, such as yarn breaks, holes, stains, etc.

2) BSData: The BSData [37] data set consists of 1,104 micrographs of the surface of ballscrews captured on the precision drive component production line. Different categories of defects are presented, such as pockmarks and pitting.

3) CFD: As a pavement crack data set, the CFD [38] data set includes 118 grayscale images of urban pavements, which cover transverse, longitudinal and alligator patterns under variable lighting, shadows and texture clutter.

4) Crack500: The Crack500 [39] data set contains 500 images of pavement cracks, captured on urban roads, including longitudinal and transverse cracks and web cracks.

5) CrackTree200: The CrackTree200 [40] data set is a challenging pavement crack data set, which contains 200 pavement images captured under varying conditions, such as shaded, occluded, low-contrast and other complex conditions.

6) DeepCrack: The DeepCrack data set [41] comprises 537 images, spanning concrete and asphalt scenes, with three surface textures, i.e., bare, dirty and rough. The cracks presented in this data set range from hairline to centimetre-wide.

7) Eugen_Miller: The Eugen_Miller [42] data set contains 55 images with a blue cement wall as the background. The defects mainly include single cracks, multiple intersecting cracks and other categories.

8) VisA: The VisA [43] data set has a total of 10,821 images, divided into 12 categories of industrial parts. e.g., PCBs and capsules. These images cover more than 40 categories of defects, such as scratches, pits, and discoloration.

9) GAPs: The GAPs [44] data set contains 2,975 images of German asphalt pavements collected from highway/urban pavements with three types of defects, including cracks, potholes and patches.

10) KolektorSDD: The KolektorSDD [45] data set consists of 399 commutator images with different categories of defects, such as scratches, ablations, etc., captured on the Kolektor production line.

11) KolektorSDD2: The KolektorSDD2 [45] data set contains 3,361 high-resolution commutator images, in which the number of defect categories is increased to eight.

12) LIACI: The LIACI [46] data set contains 1,893 images of underwater hulls. Different categories of defects are included, such as cracks, corrosion, scratches and paint flaking.

13) MT: The MT [47] data set consists of images of six common categories of tile defects, including blowhole, break, crack, fray, uneven and free (i.e., normal). In total, 1,344 images were captured from industrial magnetic-tile production lines under uniform backlighting.

14) MVTec-AD: The MVTec-AD [48] data set consists of 5,354 images which cover five textures and ten objects. This data set contains high-resolution images of 73 categories of defects across a wide range of domains.

15) NEU: The NEU [49] data set comprises 1,800 images of hot rolled steel strips. This data set covers six types of defects, such as rolled scales, plaques, cracks, etc.

16) OUC-Crack: The OUC-Crack [50] data set was captured from the cracks on the wall surface using a drone at the Laoshan Campus of Ocean University of China. This data set contains 1,968 crack images in total.

17) Rissbilder: The Rissbilder [51] data set contains 1,500 images of road cracks, collected from German pavements.

18) RubberTires: The RubberTires data set includes 1,324 X-ray images of different categories of defects of rubber tires, such as bubbles, foreign objects, etc.

19) RSDD: The RSDD [52] data set comprises 67 surface crack images captured from express rails.

20) RSDD2: The RSDD2 [52] data set was acquired from the surface of common/heavy haul rails. This data set consists of 128 crack images.

21) Volker: The Volker [51] data set comprises 108 concrete surface images sourced from bridge decks and road surfaces. In total, three categories of defects, including cracks, spalling and corrosion, are included.

B. Textual Descriptions

The 21 data sets were merged into a single data set. We manually annotated each of the 35,741 images using a textual description. To be specific, each defective image in a data set was annotated with a specific defect category, such as bubble, crack, foreign object, scratch, etc. The category information is useful for the model to identify and distinguish different categories of defects. Both the location and color of defects are also included in the textual descriptions. The location data is able to guide the model toward localizing the defect, while the color information offers an additional cue associated with the appearance of the defect. In addition, some more detailed information, such as the shape, size and texture of defects are described. These rich contextual information of defects allow the model to identify and understand defects in a more comprehensive manner. Fig. 4 shows eight different defective images and the associated textual descriptions.

The textual annotations contained in the TADD can serve dual purposes. First, these descriptions provide rich contextual information for multi-modal feature learning and alignment. Second, they can act as the ground-truth data for training an LLM in order to generate defect descriptions. Therefore, the model trained using the TADD is able to learn complicated characteristics and patterns of defects. In addition, the TADD covers various categories of defects that occur with different samples. In this case, the model tends to demonstrate strong generalizability to a wide range of defect detection scenarios. As a result, the model can achieve high accuracy and robustness in defect detection and description tasks.

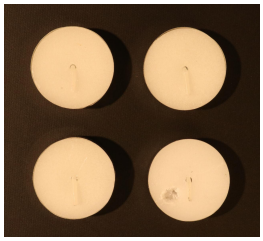
V. Experimental Setup

In this section, we first introduce the baseline methods, data sets and evaluation metrics used in this study. Then we describe the implementation details of our experiments.

A. Baselines

In quantitative evaluation experiments, we utilized ten state-of-the-art baseline methods, including SPADE [53], PaDiM [54], PatchCore [55], WinCLIP [1], PromptAD [33], AnomalyGPT [2], AnomalyCLIP [32], AA-CLIP [7], AdaCLIP [6] and Myriad [56]. These methods cover three typical categories, including classical deep defect detection methods, the image-text contrastive learning-based defect detection methods and the LVLM-based defect detection methods. Within the qualitative analysis, the proposed SPGDD-GPT was compared with three LLM-based methods, including AnomalyGPT [2], MiniGPT-4 [57] and PandaGPT [23].

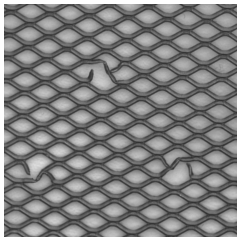
This is a picture of **a capsule placed on a white background**. The capsule is **half black and half red with the number 500 printed in white on the red side**. There is a long white crack in the top of the red area of the capsule of this picture.



This is **a top-down view of an image containing four white candles**. There is a damaged area in the lower-right surface of the candle of this picture.

This is a picture of a **brownish-yellow wooden board with light black stripes**. There are four diagonal white scratches in the right half of the board of this picture.

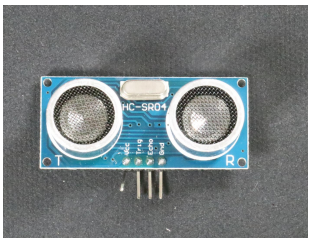
This is a picture of **a crescent-shaped cashew nut that is yellow in color**. There is a small white damage in the upper-left corner of the cashew of this picture.



This is a picture of a **black grid diagram**. There are three breaks in the top-centre, bottom-left and bottom-right of the grid of this picture.

This is a picture of **a black zipper**. There is a small portion of the zipper's teeth is broken in the bottom of the zipper of this picture.

This is a picture of **a white piece of chewing gum**. There is a crater-shaped break in the far left of the white chewing gum of this picture.



This is a picture of **a pcb board**. This **pcb board has 4 tentacles**. There is a bent defect in the leftmost tentacle of the PCB board of this picture.

Fig. 4: Examples of the textual descriptions contained in our Text-Augmented Defect Data Set (TADD). The bold fonts provide a general description of the image. The blue fonts indicate the category of defects, while the red fonts suggest the location of defects. In addition, the underlined fonts indicate more detailed information of defects.

To ensure a fair comparison, all baselines were retrained on the TADD data set. Regarding the classical methods (e.g., SPADE [53], PaDiM [54] and PatchCore [55]), we directly employed their original implementations with default hyper-parameters and parameter freezing strategies. In terms of the LVLM-based methods (e.g., WinCLIP [1], PromptAD [33], AnomalyGPT [2], Myriad [56] and AA-CLIP [7]), the official codebases and parameter freezing strategies were utilized. For the methods which used a few-shot setting in the original publication, we strictly adhered to the original configurations. When the methods (e.g., AnomalyCLIP [32] and AdaCLIP [6]) did not address few-shot learning, we followed the setup described in [2]. Since LVLM methods (e.g., AnomalyGPT and Myriad) required fine-tuning, we utilized their original fine-tuning strategies and configurations. All baselines used the Adam optimizer with a learning rate of 1e-4. They were trained at a batch size of 8 for 50 epochs.

B. Data Sets

The zero-shot defect detection experiment was conducted on the MVTec-AD [48] and VisA [43] data sets, following previous studies [1, 2, 32, 33]. The MVTec-AD [48] data set contains 3,629 normal images and 1,725 abnormal images, divided into 15 categories. In contrast, the VisA [43] data set comprises 8,821 normal images and 1,200 abnormal images, distributed in 12 categories. We used the same partitioning strategy and random seed ($SEED = 42$) as those used in the previous studies [1, 2, 32, 33] for the two data sets.

On the other hand, the few-shot defect detection experiment was performed on the TADD, which comprises approximately 16,000 normal images and 19,500 abnormal images, divided into more than 30 categories. We split the TADD into the training, validation and testing sets, which contained 70%, 15% and 15% of the TADD, respectively. The splitting was performed per category of defects in terms of each of the 21 data sets. This strategy ensured that the testing set covered all categories of defects. It should be noted that at least 10 samples were retained in the testing set, to ensure the reliability of evaluation for the categories which contained less than 50 images. A fixed random seed ($SEED = 42$) was used to guarantee experimental reproducibility.

C. Evaluation Metrics

Following the existing study [2], the Area Under the Receiver Operation Characteristic (AUROC) or the Area Under the Curve (AUC) metric was used to measure the performance of defect detection tasks. In terms of the image-level and pixel-level tasks, the metric is referred to as Image-AUC and Pixel-AUC, respectively. In addition, we assessed the performance of our method by reporting the average accuracy computed at the image-level based on the text responses of the LVLM. This metric can serve as an important indicator of the effectiveness of the LVLM-based defect detection methods. Therefore, we use Image-AUC and Pixel-AUC to evaluate the defect detection capability of the model, while evaluating

TABLE I: The Image-AUC and Pixel-AUC values obtained using ten baselines and our method in the zero-shot defect detection experiment. Here, the data sets located at the left and right sides of the arrow "→" denote the training and testing data sets, respectively. In terms of each metric, the best result is highlighted in the bold fonts. This continues for the following tables.

Method	VisA [43] → MVTec-AD [48]		MVTec-AD [48] → VisA [43]	
	Image-AUC	Pixel-AUC	Image-AUC	Pixel-AUC
SPADE [53]	77.9	80.4	75.3	77.8
PaDiM [54]	73.8	75.2	59.1	60.4
PatchCore [55]	79.5	82.8	73.8	76.7
WinCLIP [1]	91.8	85.1	76.4	78.1
PromptAD [33]	81.8	83.2	74.6	87.5
AnomalyGPT [2]	93.2	94.6	85.1	93.8
Myraid [56]	93.2	94.6	85.1	93.8
AnomalyCLIP [32]	91.5	91.1	82.1	95.4
AA-CLIP [7]	90.5	91.9	84.6	**95.5**
AdaCLIP [6]	90.0	89.9	84.3	**95.5**
SPGDD-GPT (Ours)	**93.6**	**94.8**	**85.5**	94.2

the textual description capability of the LLM using average accuracy. With the use of these metrics, the overall defect diagnosis capability of the model can be evaluated.

D. Implementation Details

Regarding the LLM utilized during the inference stage, we used the pre-trained Vicuna-7B [31] model. Both the CLIP and and ViT-B/16+ models pre-trained on the LAION-400M [58] data set were utilized, following the study of WinCLIP [1]. We set the resolution of images to 240×240 pixels. The learning rate and batch size were set to 1e-4 and 8, respectively. The network was trained for 50 epochs. Both the linear warm-up and single-cycle cosine learning rate decay strategies were used during the training stage. All experiments were conducted on an RTX 4090 Graphics Processing Unit (GPU).

VI. Experimental Results

In this section, we first present the results obtained in the quantitative evaluation experiments. Then we report the results derived in the qualitative analysis experiment. Finally, the results of the ablation studies are described.

A. Quantitative Evaluation

In this subsection, we report the results that we derived in the zero-shot and few-shot defect detection experiments.

1) Zero-Shot Defect Detection: In the scenario of zero-shot defect detection, we compared the proposed method with ten baselines. Two experiments were conducted on the MVTec-AD [48] and the VisA [43] data sets, respectively. In particular, a network was trained on the VisA [43] data set when the model trained was tested on the MVTec-AD [48] data set, and vice versa. The results are presented in Table I. It can be seen that SPGDD-GPT achieved better, or at least competitive, performance, compared to its counterparts. Specifically, it derived the highest Image-AUC and Pixel-AUC values on the MVTec-AD data set. Given that the VisA data set was used, the Image-AUC value that our method derived ranked the first while the Pixel-AUC value was relatively lower than those produced by AA-CLIP [7] and AdaCLIP [6].

2) Few-Shot Defect Detection: In this experiment, the proposed SPGDD-GPT and baselines were trained on our TADD and tested on each of the 21 data sets. The performances of defect detection and defect description were measured using the Pixel-AUC and accuracy metrics, respectively. The experiment was performed using three few-shot settings, including 1-shot, 2-shot and 4-shot. The Pixel-AUC values produced by our method and the ten baselines in the three settings are reported in Table II. As can be seen, our SPGDD-GPT normally produced the best result on each data set, regardless of which setting was utilized. This observation highlights the robust generalizability of our method in the scenario of few-shot defect detection.

In addition, we adopted the method that Gu et al. [2] proposed in order to evaluate the accuracy of defect description. Following the existing study [2], only the MVTec-AD [48] and VisA [43] data sets were used for testing. As shown in Table III, our method achieved the accuracy values of 90.1%, 89.0%, and 88.7% on the MVTec-AD data set under the 1-shot, 2-shot and 4-shot settings, which outperformed the second-best method by 2.7%, 3.6% and 3.4%, respectively. When the VisA data set was used, the accuracy values derived using our method were 80.6%, 79.5% and 78.9% under the three settings, which exceeded the accuracy values obtained using the second-best method by 0.1%, 0.6% and 0.6%, respectively. These results indicate that our method can generate proper textual descriptions of defects in different few-shot settings.

B. Qualitative Analysis

We used an LLM-as-a-Judge scheme to assess the quality of the textual descriptions that three state-of-the-art baselines, including PandaGPT [23], MiniGPT-4 [57] and AnomalyGPT [2], and our SPGDD-GPT generated on each TADD image in the zero-shot setting. They were fed into GPT-4V [3], together with the input image. In particular, GPT-4V was instructed to produce a score, ranging from 0 to 100, for a pair of textual description and image based on four criteria, including correctness, completeness, fine-grained details (e.g., position, color, shape, category, etc) and interpretability. If there was not a defect, the sufficiency of the description of normal content was additionally evaluated.

Using such an LLM-as-a-Judge scheme, we converted the subjective assessment of "good" or "bad" into reproducible and statistically meaningful numerical indicators, significantly expanding the objectivity of qualitative analysis without extra human efforts. The scores were averaged across all textual descriptions generated by a method. The average scores computed for PandaGPT, MiniGPT-4, AnomalyGPT and SPGDD-GPT were 35, 41.5, 83.5 and 94, respectively. It was demonstrated that our SPGDD-GPT produced more accurate and informative textural descriptions, compared to its counterparts.

In Fig. 5, the textual descriptions generated using those methods in terms of an image of the abnormal fabric are compared. As can be seen, PandaGPT [23] only determined whether or not a defect was present but could not understand

TABLE II: The Pixel-AUC values derived using ten baselines and our method on the TADD in the few-shot defect detection experiment. The experiment was performed for five runs and the average Pixel-AUC value was computed across these runs.

Setup	Method	Data Set																				
		Aitex	BSData	RubberTires	CFD	Crack500	CrackTree200	DeepCrack	Eugen_Miller	GAPs	KolektorSDD	KolektorSDD2	LIACI	MT	MVTec-AD	NEU	OUCCrack	Rissbilder	RSDD	RSDD2	Visa	Volker
1-Shot	SPADE [53]	48.6	56.8	66.3	64.9	55.0	48.8	62.6	73.7	50.7	54.7	61.8	51.2	63.2	91.2	57.3	54.9	66.3	**87.5**	83.4	95.6	59.2
	PaDiM [54]	68.8	63.7	70.5	74.0	71.3	64.6	82.1	80.2	62.9	79.5	71.8	61.9	85.7	89.3	89.3	68.8	70.5	81.7	78.0	89.9	65.4
	PatchCore [55]	63.6	50.6	59.7	87.5	65.8	80.2	66.5	82.5	55.3	74.8	85.8	50.8	83.2	92.0	62.3	57.9	59.7	60.8	54.7	95.4	54.1
	WinCLIP [1]	92.8	75.6	80.4	90.0	74.5	86.4	82.8	87.2	75.0	89.7	87.6	77.3	**92.8**	95.2	91.5	55.5	80.4	84.4	70.8	96.4	83.2
	PromptAD [33]	69.1	74.8	71.6	93.2	83.8	89.7	86.7	92.8	85.6	90.8	87.7	63.9	92.4	85.9	60.9	57.3	71.6	71.4	61.1	89.0	84.9
	AnomalyGPT [2]	85.2	80.3	89.7	94.0	86.8	90.6	89.9	92.6	96.4	71.3	70.4	82.5	84.6	95.3	**95.6**	90.0	89.7	72.2	70.3	96.2	90.5
	Myraid [56]	85.2	80.3	89.7	94.0	86.8	90.6	89.9	92.6	96.4	71.3	70.4	82.5	84.6	95.3	**95.6**	90.0	89.7	72.2	70.3	96.2	90.5
	AnomalyCLIP [32]	82.1	79.3	87.2	93.1	85.5	89.2	80.4	91.7	96.0	87.2	86.5	62.4	83.2	90.6	78.8	65.5	87.2	71.0	69.4	89.8	89.8
	AA-CLIP [7]	78.8	76.8	86.9	92.8	85.1	88.9	88.0	91.3	95.5	69.8	69.1	80.9	82.8	89.9	94.8	76.4	76.8	70.6	69.0	91.9	79.5
	AdaCLIP [6]	73.4	80.6	76.3	89.3	84.2	88.5	79.0	89.1	96.4	88.6	87.2	81.7	84.0	87.6	83.2	60.1	74.3	71.5	69.8	81.2	85.6
	SPGDD-GPT (Ours)	**93.2**	**85.6**	**93.4**	**94.8**	**87.1**	**91.5**	**90.5**	**94.5**	**97.5**	**91.0**	**88.6**	**86.7**	88.8	**95.8**	**95.6**	**91.8**	**93.4**	85.8	**83.9**	**96.8**	**91.0**
2-Shot	SPADE [53]	59.8	52.6	55.8	66.0	55.9	50.0	63.0	75.2	52.3	56.4	62.0	58.7	63.1	92.0	60.1	56.7	55.8	**89.1**	83.8	96.2	62.3
	PaDiM [54]	69.3	68.4	73.2	74.5	72.9	64.5	82.9	82.2	63.1	82.1	73.2	64.6	86.0	91.3	89.8	80.2	73.2	81.8	77.9	92.0	65.4
	PatchCore [55]	65.2	58.7	65.9	87.5	67.0	81.4	67.0	82.4	58.2	76.3	86.9	60.4	83.7	93.3	63.0	58.4	57.2	62.4	55.2	96.1	55.8
	WinCLIP [1]	92.9	68.8	74.8	91.3	74.8	87.0	83.5	88.6	76.6	90.6	89.1	70.5	**93.8**	**96.0**	93.2	65.9	74.8	85.2	72.2	96.8	84.3
	PromptAD [33]	71.4	72.6	81.4	94.8	84.0	91.9	88.5	93.0	85.8	90.8	88.4	75.6	93.2	86.4	62.7	57.9	81.4	73.6	63.0	89.8	85.7
	AnomalyGPT [2]	87.2	84.6	91.5	95.8	88.1	91.9	91.2	93.9	97.5	72.6	72.4	85.3	86.7	95.6	96.0	92.3	91.5	74.8	72.0	96.4	91.6
	Myraid [56]	87.2	84.6	91.5	95.8	88.1	91.9	91.2	93.9	97.5	72.6	72.4	85.3	86.7	95.6	96.0	92.3	91.5	74.8	72.0	96.4	91.6
	AnomalyCLIP [32]	83.4	81.2	88.9	94.5	87.1	90.7	81.0	92.9	97.2	87.6	87.2	65.1	84.7	90.8	80.2	66.1	88.5	72.5	70.6	90.2	90.4
	AA-CLIP [7]	80.1	78.4	88.3	94.0	86.6	90.2	89.5	92.4	96.8	71.3	70.7	82.6	84.2	91.0	95.0	76.8	78.0	72.0	70.1	93.4	80.9
	AdaCLIP [6]	73.6	80.1	77.8	90.5	85.4	89.6	80.5	90.3	97.4	89.3	88.4	83.7	85.6	88.9	84.6	62.6	74.5	73.6	71.4	80.9	85.9
	SPGDD-GPT (Ours)	**93.6**	**88.5**	**93.7**	**96.3**	**89.5**	**92.7**	**91.8**	**95.0**	**97.8**	**92.4**	**90.3**	**89.4**	90.8	95.9	**96.7**	**93.2**	**93.7**	87.0	**85.6**	**97.2**	**93.8**
4-Shot	SPADE [53]	60.6	54.2	57.4	66.4	57.3	51.7	64.1	75.8	53.2	57.9	63.2	60.1	65.0	92.7	62.0	58.4	57.4	**91.8**	84.5	96.6	63.9
	PaDiM [54]	71.7	69.5	74.8	75.6	74.3	64.8	84.2	84.7	64.4	84.0	74.6	66.3	87.0	92.6	90.3	82.5	74.8	83.0	79.0	93.2	66.1
	PatchCore [55]	66.8	59.8	67.5	88.3	68.3	81.8	67.9	84.3	60.6	76.8	87.4	61.6	84.9	94.3	64.5	60.1	67.5	63.8	56.5	96.8	57.4
	WinCLIP [1]	93.4	69.6	75.4	93.0	76.2	87.8	83.8	89.7	77.3	91.6	90.7	71.4	**95.7**	96.2	94.5	57.9	75.4	86.5	73.4	97.2	85.6
	PromptAD [33]	73.7	74.5	83.1	96.0	85.5	92.1	90.3	93.5	86.4	91.5	89.7	76.8	94.8	87.4	63.6	59.5	83.1	74.6	64.5	90.3	87.0
	AnomalyGPT [2]	88.4	85.4	92.8	96.5	89.2	93.0	92.6	94.1	98.0	74.3	74.5	86.9	87.2	96.2	96.8	93.4	92.8	76.4	73.6	96.7	92.4
	Myraid [56]	88.4	85.4	92.8	96.5	89.2	93.0	92.6	94.1	98.0	74.3	74.5	86.9	87.2	96.2	96.8	93.4	92.8	76.4	73.6	96.7	92.4
	AnomalyCLIP [32]	85.3	82.4	90.1	95.0	88.9	92.2	82.0	93.8	97.6	89.5	88.4	70.4	87.1	92.4	81.6	69.4	90.2	75.1	73.0	91.3	90.8
	AA-CLIP [7]	80.6	81.7	89.6	95.3	88.3	92.0	91.1	92.6	97.0	73.0	72.5	84.7	86.6	91.1	95.2	78.5	80.1	74.6	72.5	93.6	81.2
	AdaCLIP [6]	76.2	83.1	79.9	92.3	86.9	91.4	82.8	90.8	98.0	90.3	89.1	86.0	87.9	89.8	86.5	65.9	75.6	76.2	74.0	83.5	86.6
	SPGDD-GPT (Ours)	**94.9**	**89.8**	**94.5**	**97.4**	**90.3**	**93.7**	**93.2**	**95.4**	**98.4**	**92.8**	**92.6**	**90.5**	92.6	**96.8**	**97.3**	**94.7**	**94.5**	89.4	**87.2**	**97.3**	**94.1**

the content of the image or the defect; MiniGPT-4 [57] did not work; and AnomalyGPT [2] recognized the defect and pointed out its location, but failed to describe the attributes of the defect. In contrast, our SPGDD-GPT not only identified the defect but also described it in detail.

On the other hand, the textual descriptions generated using the four methods with regard to the image of a normal pill are compared in Fig. 6. It can be seen that PandaGPT [23] detected a defect incorrectly and misunderstood the content

TABLE III: The accuracy values obtained on the MVTec-AD [48] and VisA VisA [43] data sets using different few-shot setups.

Setup	Method	MVTec-AD	VisA
1-Shot	AnomalyGPT [2]	86.1 ± 1.1	77.4 ± 1.0
	Myraid [56]	87.4 ± 0.9	80.5 ± 1.2
	SPGDD-GPT (Ours)	**90.1 ± 1.9**	**80.6 ± 1.7**
2-Shot	AnomalyGPT [2]	84.8 ± 0.8	77.5 ± 0.3
	Myraid [56]	85.4 ± 0.7	78.9 ± 0.5
	SPGDD-GPT (Ours)	**89.0 ± 2.3**	**79.5 ± 1.6**
4-Shot	AnomalyGPT [2]	85.0 ± 0.3	77.7 ± 0.4
	Myraid [56]	85.3 ± 0.3	78.3 ± 0.3
	SPGDD-GPT (Ours)	**88.7 ± 2.3**	**78.9 ± 1.7**

of the image, while MiniGPT-4 [57] correctly identified the absence of a defect but inaccurately understood the image. Although AnomalyGPT [2] correctly determined there was not a defect, it described the image roughly. Compared with these baselines, the proposed SPGDD-GPT confirmed the absence of a defect and described the content of the image in detail.

C. Ablation Studies

To investigate the effectiveness of different modules in the proposed network, we conducted a series of ablation experiments. Specifically, we examined the impact of the Text-Driven Defect Focuser (TDDF), the Multi-scale Self-prompted Memory Module (MSSPMM), the Adapter Module and the Enhanced VV-CLIP. In addition, we compared two different fine-tuning approaches for the Vicuna [31] model. For simplicity, the network was trained using our TADD but was tested on the MVTec-AD [48] and VisA [43] data sets under the 1-shot setting.

1) Impact of the Text-Driven Defect Focuser (TDDF): We obtained a variant (w/o TDDF) of the SPGDD-GPT by replacing the TDDF with a simple image and text embedding multiplication operation. In essence, cosine similarity and the standard contrastive loss of CLIP [8] were used in this case. A second variant (Only TDDF) was derived by removing both the MSSPMM and the Adapter Module from the SPGDD-GPT. Without the Adapter Module, this variant could not generate a textual description, and thus the accuracy metric was not applicable. We compared the results produced by the SPGDD-GPT (w/ TDDF) with those derived using the two variants in Table IV. It can be seen that our SPGDD-GPT performed much better using the TDDF than it did without the TDDF or only using the TDDF, no matter which metric was considered across the two data sets. This finding highlights the important role of the TDDF in focusing the model on defective regions.

2) Impact of the Multi-scale Self-prompted Memory Module (MSSPMM): In Table V, we compare the results derived using our SPGDD-GPT with and without the MSSPMM. As can be observed, the use of the MSSPMM improved the performance of the SPGDD-GPT. In particular, the MSSPMM boosted the Image-AUC, Pixel-AUC and accuracy values by 2.6%, 2.5% and 1.1% on the MVTec-AD [48] data set and by 2.4%, 3.3% and 1.8% on the VisA [43] data set, respectively. This

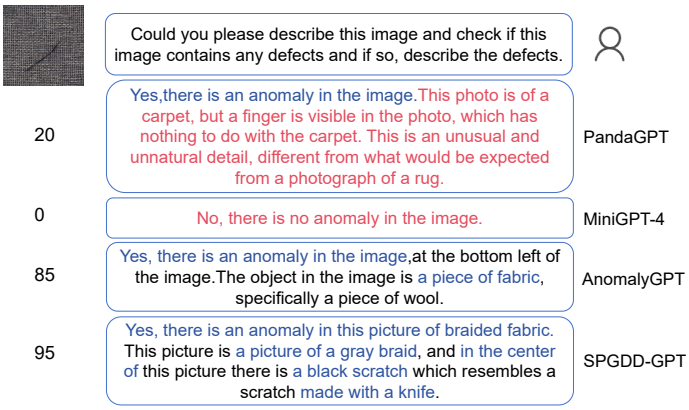


Fig. 5: Comparison of the textual descriptions generated using PandaGPT [23], MiniGPT-4 [57], AnomalyGPT [2] and our SPGDD-GPT with regard to a fabric image with a scratch. A score generated by GPT-4V [3] is shown at the left side of each textual description. Compared to the three baselines, our SPGDD-GPT not only identifies and localizes the defect but also produces the detailed description of the color, shape and additional attributes of the defect.

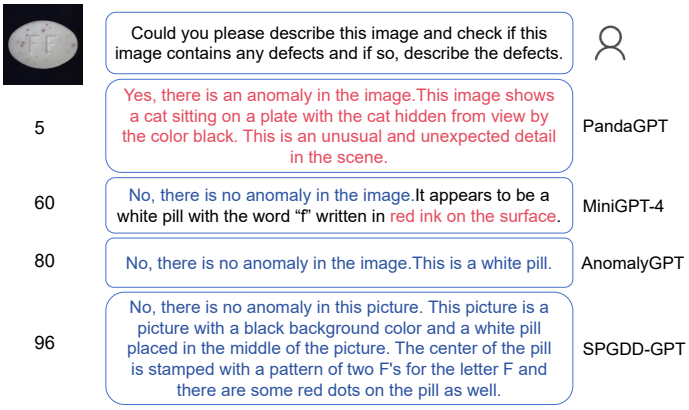


Fig. 6: Comparison of the textual descriptions generated using PandaGPT [23], MiniGPT-4 [57], AnomalyGPT [2] and our SPGDD-GPT in terms of a normal image which contains a pill. A score generated by GPT-4V [3] is shown at the left side of each textual description. In contrast to the three baselines, our SPGDD-GPT not only correctly determines the absence of defects but also generates the detailed characterization of the image content.

TABLE IV: Comparison between the Image-AUC, Pixel-AUC and accuracy (Acc.) values obtained using the SPGDD-GPT with and without the TDDF and only with the TDDF on the MVTec-AD [48] and VisA [43] data sets.

Method	MVTec-AD			VisA		
	Image-AUC	Pixel-AUC	Acc.	Image-AUC	Pixel-AUC	Acc.
w/o TDDF	87.4	88.6	86.5	81.7	83.6	76.7
w/ TDDF	**95.2**	**95.8**	**90.1**	**88.5**	**96.8**	**80.6**
Only TDDF	92.9	94.0	-	86.0	93.5	-

observation should be due to the ability of the MSSPMM to focus on abnormal regions by capturing contextual information

TABLE V: Comparison between the Image-AUC, Pixel-AUC and accuracy (ACC.) values obtained using the SPGDD-GPT with and without the MSSPMM on the MVTec-AD [48] and VisA [43] data sets.

Method	MVTec-AD			VisA		
	Image-AUC	Pixel-AUC	Acc.	Image-AUC	Pixel-AUC	Acc.
w/o MSSPMM	92.6	93.3	89.0	86.1	93.5	78.8
w/ MSSPMM	**95.2**	**95.8**	**90.1**	**88.5**	**96.8**	**80.6**

TABLE VI: Comparison between the Image-AUC, Pixel-AUC and accuracy (ACC.) values obtained using the SPGDD-GPT with and without the Adapter Module on the MVTec-AD [48] and VisA [43] data sets.

Method	MVTec-AD			VisA		
	Image-AUC	Pixel-AUC	Acc.	Image-AUC	Pixel-AUC	Acc.
w/o Adapter	94.8	95.1	89.4	87.6	95.7	79.5
w/ Adapter	**95.2**	**95.8**	**90.1**	**88.5**	**96.8**	**80.6**

TABLE VII: Comparison between the Image-AUC, Pixel-AUC and accuracy (ACC.) values derived using the SPGDD-GPT with different image encoders on the MVTec-AD [48] and VisA [43] data sets.

Image Encoder	MVTec-AD			VisA		
	Image-AUC	Pixel-AUC	Acc.	Image-AUC	Pixel-AUC	Acc.
CLIP [8]	75.2	76.9	70.5	75.9	77.5	64.2
MaskCLIP [59]	91.2	92.4	89.5	**90.6**	92.8	80.1
Enhanced VV-CLIP	**95.2**	**95.8**	**90.1**	88.5	**96.8**	**80.6**

and using the result of zero-shot defect detection.

3) Impact of the Adapter Module: We examined the effect of the Adapter Module by comparing the results produced by the SPGDD-GPT with and without this module in Table VI. As can be seen, the utilization of the Adapter Module improved the performance of our method on both the MVTec-AD [48] and VisA [43] data sets. It is indicated that the Adapter Module not only optimized the accuracy of text generation but also enhanced the ability of the model to detect defects.

4) Impact of the Enhanced VV-CLIP: In addition to the Enhanced VV-CLIP that we adopted, we tested two different image encoders, including CLIP [8] and MaskCLIP [59], together with the proposed SPGDD-GPT. As reported in Table VII, our Enhanced VV-CLIP normally produced the better result on both the MVTec-AD [48] and VisA [43] data sets, compared with its counterparts. This finding should be due to the stronger feature representation ability of the Enhanced VV-CLIP than those of CLIP and MaskCLIP, which also improved the performance of defect description.

5) Impact of the Fine-Tuning Approach: We also applied LoRA [35] to fine-tuning the Vicuna [31] model for comparison purposes. The results produced by our SPGDD-GPT using two fine-tuning methods are reported in Table VIII. It can be observed that the model fine-tuned using the Adapter Module always outperformed that fine-tuned using LoRA across the two data sets. By referring to Table VI, we can find that the model that has not been fine-tuned produced the same

TABLE VIII: Comparison between the Image-AUC, Pixel-AUC and accuracy (ACC.) values derived using the SPGDD-GPT with different fine-tuning approaches on the MVTec-AD [48] and VisA [43] data sets.

Method	MVTec-AD			VisA		
	Image-AUC	Pixel-AUC	Acc.	Image-AUC	Pixel-AUC	Acc.
LoRA	94.8	95.1	86.4	87.6	95.7	75.2
Adapter	**95.2**	**95.8**	**90.1**	**88.5**	**96.8**	**80.6**

Image-AUC and Pixel-AUC values as those produced by the model fine-tuned using LoRA. This finding is due to the fact that the fine-tuning operation does not affect the performance of defect detection. However, the model fine-tuned using LoRA achieved the worse defect description result (Acc.) than that produced using the model without fine-tuning. These results demonstrate that the Adapter Module is useful for improving the performance of the model for defect description by enhancing the adaptability of the model to various defects.

D. Failure Analysis

Although the proposed SPGDD-GPT has achieved promising performance, it still exhibits two limitations. One limitation arises with sub-patch-scale defects. For instance, a microscopic scratch (approximately 2 pixels or 0.2 mm in width) occurs on a PCB (see Fig. 7(a-b)). In this case, the defective area is much smaller than the 16×16 patch used in the local processing branch of our Enhanced VV-CLIP while the local self-attention mechanism likely fails to capture a response from the patch containing the defect. As a result, our model will miss the scratch. A second limitation will occur when the color or intensity of the defects is nearly identical to the substrate material (see Fig. 7(c-d)). In such a scenario, the color- or intensity-related textual embeddings generated by the text encoder of CLIP provide a weak or non-discriminative representation. This prevents the TDDF from effectively separating the defect from the background based on the color or intensity cue, resulting in confusion and the failure to identify the defective region. The two limitations reveal the dependencies of our method on the granularity of the image encoder and the discriminative power of text encoder for fine-grained visual attributes, respectively.

E. Statistical Significance Analysis

To validate the significance of the performance gain that our method achieved, we performed a paired *t*-test between the results derived using our method on the 21 data sets and those produced by the best baseline on each data set. Given that the Pixel-AUC values reported in Table II were used, SPGDD-GPT achieved statistically significant advantages across all few-shot settings. Specifically, the performance gains are +1.37% (p = 0.019), +1.40% (p = 0.004) and +1.70% (p = 0.00033) in the 1-shot, 2-shot and 4-shot settings, respectively. The consistently significant results ($p < 0.05$ to $p < 0.001$), coupled with the superior performance on 15 to 18 out of the 21 data sets, demonstrate the effectiveness and robustness of our method.

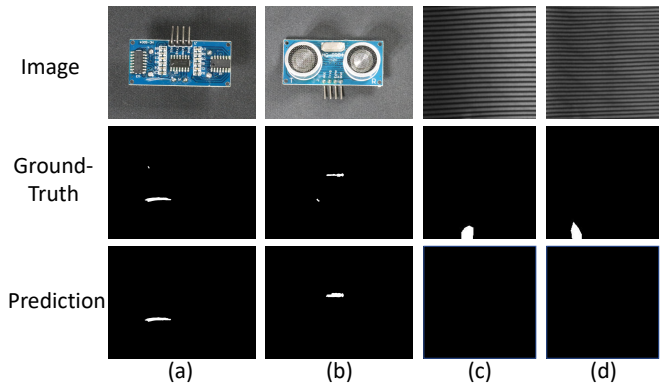


Fig. 7: Examples of typical failure cases produced by the proposed SPGDD-GPT.

VII. Conclusion

In this paper, we constructed a large-scale defect data set, namely, the Text-Augmented Defect Data Set (TADD). This data set contained 35,741 images across 21 defect subsets, together with the detailed textual descriptions that we annotated, which provided rich contextual information for training a Large Vision-Language Model (LVLM). In addition, we proposed a Self-prompted Generic Defect Diagnosis (including defect detection and defect description) LVLM, i.e., the SPGDD-GPT. Specifically, we deliberately designed a Text-Driven Defect Focuser (TDDF) and a Multi-scale Self-prompted Memory Module (MSSPMM). The TDDF leveraged textual descriptions to focus the model on defective regions by optimizing the Euclidean distance between defective image features and text embeddings, while the MSSPMM used a small number of normal samples as memory to help the model quickly adapt to unseen defects and focus on defective regions. Experimental results demonstrated that the SPGDD-GPT normally achieved the better defect detection and defect description performances, compared to its counterparts. We believe that the promising results are due to the large-scale TADD and the proposed MSSPMM and TDDF.

References

[1] J. Jeong, Y. Zou, T. Kim, D. Zhang, A. Ravichandran, and O. Dabeer, "Winclip: Zero-/few-shot anomaly classification and segmentation," *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19 606–19 616, 2023.

[2] Z. Gu, B. Zhu, G. Zhu, Y. Chen, M. Tang, and J. Wang, "Anomalygpt: Detecting industrial anomalies using large vision-language models," in *AAAI Conference on Artificial Intelligence*, 2023.

[3] Z. Yang, L. Li, K. Lin, J. Wang, C.-C. Lin, Z. Liu, and L. Wang, "The dawn of lmms: Preliminary explorations with gpt-4v(ision)," *ArXiv*, vol. abs/2309.17421, 2023.

[4] Y. Li, W. Zhao, and J. Pan, "Deformable patterned fabric defect detection with fisher criterion-based deep learning," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 1256–1264, 2017.

[5] W. Zhou, J. Yang, W. Yan, and M. Fang, "Rdnet-kd: Recursive encoder, bimodal screening fusion, and knowledge distillation network for rail defect detection," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 2031–2040, 2025.

[6] Y. Cao, J. Zhang, L. Frittoli, Y. Cheng, W. Shen, and G. Boracchi, "Adaclip: Adapting clip with hybrid learnable prompts for zero-shot anomaly detection," *ArXiv*, vol. abs/2407.15795, 2024.

[7] W. Ma, X. Zhang, Q. Yao, F. Tang, C. Wu, Y. Li, R. Yan, Z. Jiang, and S. Zhou, "Aa-clip: Enhancing zero-shot anomaly detection via anomaly-aware clip," *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4744–4754, 2025.

[8] 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 *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 8748–8763.

[9] J. Li, D. Li, S. Savarese, and S. Hoi, "BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 2023, pp. 19 730–19 742.

[10] J. Xu, S. D. Mello, S. Liu, W. Byeon, T. Breuel, J. Kautz, and X. Wang, "Groupvit: Semantic segmentation emerges from text supervision," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18 113–18 123, 2022.

[11] X. Dong, Y. Zheng, J. Bao, T. Zhang, D. Chen, H. Yang, M. Zeng, W. Zhang, L. Yuan, D. Chen, F. Wen, and N. Yu, "Maskclip: Masked self-distillation advances contrastive language-image pretraining," *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10 995–11 005, 2022.

[12] W. Liang, Y. Sun, S. Zhang, L. Bai, and J. Yang, "Smallnet: A small defects detection network for magnetic chips based on context-weighted aggregation and feature multiscale loop fusion," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 10 095–10 106, 2025.

[13] N. Dong and E. P. Xing, "Few-shot semantic segmentation with prototype learning," in *British Machine Vision Conference*, 2018.

[14] J. Zhu and G. Pang, "Toward generalist anomaly detection via in-context residual learning with few-shot sample prompts," *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17 826–17 836, 2024.

[15] K. Wang, J. H. Liew, Y. Zou, D. Zhou, and J. Feng, "Panet: Few-shot image semantic segmentation with prototype alignment," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9196–9205, 2019.

[16] X. Zhang, Y. Wei, Y. Yang, and T. Huang, "Sg-one: Similarity guidance network for one-shot semantic segmentation," *IEEE Transactions on Cybernetics*, vol. 50, pp. 3855–3865, 2018.

[17] M. Yang, P. Wu, J. Liu, and H. Feng, "Memseg: A semi-supervised method for image surface defect detection using differences and commonalities," *Eng. Appl. Artif. Intell.*, vol. 119, p. 105835, 2022.

[18] M. Deitke, C. Clark, S. Lee, R. Tripathi, Y. Yang, J. S. Park, M. Salehi, N. Muennighoff, K. Lo, L. Soldaini, J. Lu, T. Anderson, E. Bransom, K. Ehsani, H. Ngo, Y. Chen, A. Patel, M. Yatskar, C. Callison-Burch, A. Head, R. Hendrix, F. Bastani, E. VanderBilt, N. Lambert, Y. Chou, A. Chheda, J. Sparks, S. Skjonsberg, M. Schmitz, A. Sarnat, B. Bischoff, P. Walsh, C. Newell, P. Wolters, T. Gupta, K.-H. Zeng, J. Borchardt, D. Groeneveld, C. Nam, S. Lebrecht, C. Wittlif, C. Schoenick, O. Michel, R. Krishna, L. Weihs, N. A. Smith, H. Hajishirzi, R. Girshick, A. Farhadi, and A. Kembhavi, "Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models," *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 91–104, 2024.

[19] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *arXiv e-prints*, p. arXiv:1408.5882, Aug. 2014.

[20] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[21] Y. Ai and T. Ye, "Surface defect detection algorithm for pcb based on improved yolov8," in *2024 8th International Conference on Electrical, Mechanical and Computer Engineering (ICEMCE)*, 2024, pp. 1421–1425.

[22] Y. Chai, X. Yao, M. Chen, and S. Shan, "Fpfs-yolo: An insulator defect detection model integrating fasternet and an attention mechanism," *Sensors*, vol. 25, no. 13, 2025.

[23] Y. Su, T. Lan, H. Li, J. Xu, Y. Wang, and D. Cai, "Pandagpt: One model to instruction-follow them all," *ArXiv*, vol. abs/2305.16355, 2023.

[24] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds *et al.*, "Flamingo: a visual language model for few-shot learning," *Advances in neural information processing systems*, vol. 35, pp. 23 716–23 736, 2022.

[25] B. Lin, Z. Tang, Y. Ye, J. Cui, B. Zhu, P. Jin, J. Zhang, M. Ning, and L. Yuan, "Moe-llava: Mixture of experts for large vision-language models," *arXiv preprint arXiv:2401.15947*, 2024.

[26] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[27] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *Advances in neural information processing systems*, vol. 36, 2024.

[28] A. Zhang, H. Fei, Y. Yao, W. Ji, L. Li, Z. Liu, and T.-S. Chua, "Vpgtrans: Transfer visual prompt generator across llms," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[29] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma *et al.*, "Scaling instruction-finetuned language models," *Journal of Machine Learning Research*, vol. 25, no. 70, pp. 1–53, 2024.

[30] R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. V. Alwala, A. Joulin, and I. Misra, "Imagebind one embedding space to bind them all," *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15 180–15 190, 2023.

[31] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez *et al.*, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," *See https://vicuna. lmsys. org (accessed 14 April 2023)*, vol. 2, no. 3, p. 6, 2023.

[32] Q. Zhou, G. Pang, Y. Tian, S. He, and J. Chen, "Anomalyclip: Object-agnostic prompt learning for zero-shot anomaly detection," *ArXiv*, vol. abs/2310.18961, 2023.

[33] X. Li, Z. Zhang, X. Tan, C. Chen, Y. Qu, Y. Xie, and L. Ma, "Promptad: Learning prompts with only normal samples for few-shot anomaly detection," *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16 848–16 858, 2024.

[34] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, "Parameter-efficient fine-tuning for large models: A comprehensive survey," *Trans. Mach. Learn. Res.*, vol. 2024, Mar 2024.

[35] J. E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *ArXiv*, vol. abs/2106.09685, 2021.

[36] 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.

[37] T. Schlagenhauf, M. Landwehr, and J. Fleischer, "Industrial machine tool component surface defect dataset," *Data in Brief*, vol. 39, 2021.

[38] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen, "Automatic road crack detection using random structured forests," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3434–3445, 2016.

[39] F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, and H. Ling, "Feature pyramid and hierarchical boosting network for pavement crack detection," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[40] Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang, "Cracktree: Automatic crack detection from pavement images," *Pattern Recognition Letters*, vol. 33, no. 3, pp. 227–238, 2012.

[41] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang, "Deepcrack: Learning hierarchical convolutional features for crack detection," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1498–1512, 2019.

[42] E. Bianchi and M. Hebdon, "Concrete crack conglomerate dataset," Dataset, Oct 2021, dataset. Virginia Tech University Libraries, 2021.

[43] Y. Zou, J. Jeong, L. Pemula, D. Zhang, and O. Dabeer, "Spot-the-difference self-supervised pre-training for anomaly detection and segmentation," *arXiv preprint arXiv:2207.14315*, 2022.

[44] M. Eisenbach, R. Stricker, D. Seichter, K. Amende, K. Debes, M. Sesselmann, D. Ebersbach, U. Stoeckert, and H.-M. Gross, "How to get pavement distress detection ready for deep learning? a systematic approach," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 2039–2047.

[45] D. Tabernik, S. Šela, J. Skvarč, and D. Skočaj, "Segmentation-Based Deep-Learning Approach for Surface-Defect Detection," *Journal of Intelligent Manufacturing*, May 2019.

[46] M. Waszak, A. Cardaillac, B. Elvesæter, F. Rødølen, and M. Ludvigsen, "Semantic segmentation in underwater ship inspections: Benchmark and data set," *IEEE Journal of Oceanic Engineering*, vol. 48, no. 2, pp. 462–473, 2023.

[47] Y. Huang, C. Qiu, Y. Guo, X. Wang, and K. Yuan, "Surface defect saliency of magnetic tile," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, 2018, pp. 612–617.

[48] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Mvtec ad — a comprehensive real-world dataset for unsupervised anomaly detection," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9584–9592.

[49] Y. Bao, K. Song, J. Liu, Y. Wang, Y. Yan, H. Yu, and X. Li, "Triplet-graph reasoning network for few-shot metal generic surface defect segmentation," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–11, 2021.

[50] B. Xu, W. Shao, and X. Dong, "Drone-based wall crack detection using model-agnostic meta-learning," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 15 116–15 128, 2025.

[51] S. Kulkarni, S. Singh, D. Balakrishnan, S. Sharma, S. Devunuri, and S. C. R. Korlapati, "Crackseg9k: A collection and benchmark for crack segmentation datasets and frameworks," in *European Conference on Computer Vision (ECCV) Workshops*, ser. Lecture Notes in Computer Science, vol. 13807. Springer, 2022, pp. 179–195.

[52] M. Niu, K. Song, L. Huang, Q. Wang, Y. Yan, and Q. Meng, "Unsupervised saliency detection of rail surface defects using stereoscopic images," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2271–2281, 2021.

[53] N. Cohen and Y. Hoshen, "Sub-image anomaly detection with deep pyramid correspondences," 2021.

[54] T. Defard, A. Setkov, A. Loesch, and R. Audigier, "PaDiM: a Patch Distribution Modeling Framework for Anomaly Detection and Localization," *arXiv e-prints*, p. arXiv:2011.08785, Nov. 2020.

[55] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler, "Towards Total Recall in Industrial Anomaly Detection," *arXiv e-prints*, p. arXiv:2106.08265, Jun. 2021.

[56] Y. Li, H. Wang, S. Yuan, M. Liu, D. Zhao, Y. Guo, C. Xu, G. Shi, and W. Zuo, "Myriad: Large multimodal model by applying vision experts for industrial anomaly detection," *arXiv preprint arXiv:2310.19070*, 2023.

[57] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "MiniGPT-4: Enhancing Vision-Language Understanding with Advanced Large Language Models," *arXiv e-prints*, p. arXiv:2304.10592, Apr. 2023.

[58] Y. Li, H. Fan, R. Hu, C. Feichtenhofer, and K. He, "Scaling language-image pre-training via masking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 23 390–23 400.

[59] C. Zhou, C. C. Loy, and B. Dai, "Extract free dense labels from clip," in *European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science, vol. 13688. Springer, 2022, pp. 696–712.

Shengwang An received the bachelor's degree in Engineering from Zhengzhou University of Light Industry (ZZULI), Zhengzhou, Henan Province, China, in 2023. He is currently a post-graduate student at Ocean University of China working toward his master's degree in Software Engineering. His research interests include computer vision, defect detection, deep learning, and image segmentation.

Xinghui Dong received the PhD degree from Heriot-Watt University, U.K., in 2014. He worked with the Centre for Imaging Sciences, the University of Manchester, U.K., between 2015 and 2021. Then he joined Ocean University of China in 2021. He is currently a professor at the Ocean University of China. His research interests include computer vision, defect detection, texture analysis, and visual perception.