Applied Intelligence (2025) 55:716
https://doi.org/10.1007/s10489-025-06590-3

# Small sample pipeline DR defect detection based on smooth variational autoencoder and enhanced detection head faster RCNN

Ting Zhang[1] · Tianyang You[1] · Zhaoying Liu[1] · Sadaqat Ur Rehman[2] · Yanan Shi[3,4] · Amr Munshi[5]

Accepted: 22 April 2025


**Abstract**
The safe operation of gas pipelines is crucial for the safety of residents' lives and property. However, accurately detecting defects within these gas pipelines is a challenging task. To improve the accuracy of defect detection in pipeline DR images with small sample sizes, we propose an enhanced Faster RCNN model based on a Smooth Variational Autoencoder and Enhanced Detection Head (S-EDH-Faster RCNN). This model leverages a smooth variational autoencoder to reconstruct features and enhances classification scores through an improved detection head, thereby boosting overall detection accuracy. In detail, to address the issue of scarce training samples for new categories, we design a smooth variational autoencoder to reconstruct features that better fit the distribution of training data. Furthermore, to refine classification precision, we present an enhanced detection head that incorporates a convolutional block attention-based center point classification calibration module, which strengthens classification-related portions of the RoI features and adjusts classification scores accordingly. Finally, to effectively learn characteristics of novel class samples, we introduce an adaptive fine-tuning method that adaptively updates key convolutional kernels during the fine-tuning stage, enabling the model to generalize better to novel classes. Experimental results demonstrate that our approach achieves superior detection performance over state-of-the-art models on both the home-made PIP-DET dataset and the publicly available NEU-DET dataset, demonstrating its effectiveness.

**Keywords** Few-shot learning · Variational autoencoder · Enhanced detection head · Adaptive fine-tuning · Pipeline DR defect detection

## 1 Introduction

The health status of pipeline systems plays a critical role in ensuring the safety of production and operations [1, 2]. Although significant advancements have been made in automatic defect detection technologies in recent years, these methods usually rely on large amounts of labeled data to train efficient detection models. In practical applications, however, obtaining large-scale annotated defect data is often challenging and costly [3–6]. Thus, few-shot learning techniques are particularly important in the field of pipeline defect detection [7–10].

To enhance the accuracy of defect detection under few-shot conditions, researchers have proposed various methods. Based on their training paradigms, these approaches can primarily be categorized into meta-learning-based methods and fine-tuning-based methods [11–13]. The meta-learning-based methods improve the model's generalization capability for new tasks by teaching it how to learn [14, 15]. The specific process involves initially pre-training a meta-model using a large dataset, followed by rapid learning and adjustment with a small amount of labeled data in new tasks to achieve effective object detection under few-shot conditions [16, 17]. For instance, Han et al. proposed the Meta Faster RCNN [18], which constructed two detection branches to detect base class samples and novel class samples separately, thereby reducing classification confusion between base and novel classes. Han et al. introduced a few-shot object detection method via Variational Feature Aggregation (VFA) [19]. This approach, based on Faster RCNN, incorporates a variational autoencoder to model support set samples, generating a normally distributed feature distribution and then sampling from it as class prototypes to improve detection accuracy.

However, this type of methods often entail complex training processes and data organization, limiting their application. In contrast, the fine-tuning-based methods are simpler and more effective [5, 20, 21]. These methods first use a large number of base class samples for initial model training, then

Extended author information available on the last page of the article

Published online: 03 May 2025

fine-tune the detection head using a balanced dataset composed of some base class samples and all novel class samples, with the aim to achieve good detection performance on new categories [22]. For example, Wang et al. proposed a Two-stage Fine-tuning Approach (TFA) [8]. They first trained the entire Faster RCNN on a large number of base classes, then fine-tuned the detection head using novel class samples to improve detection precision. Li et al. introduced a Class Margin Equilibrium (CME) [23]. This method, also based on Faster RCNN, incorporates a feature perturbation module during the fine-tuning stage to dynamically adjust class margins, allowing novel classes to occupy appropriate positions within the feature space of base classes, thus reducing confusion between novel and different classes.

Whether employing meta-learning or fine-tuning methods, Faster RCNN has been widely used as a base detector in few-shot object detection, achieving significant performance improvements [20, 22, 24, 25]. Nevertheless, these methods still face three main issues: (1) Insufficient novel class features: Due to the limited data for novel classes, it is challenging for the model to effectively learn the features and distributions of these new categories, leading to weak generalization capabilities for novel classes; (2) Classification confusion: In few-shot scenarios, the model is prone to confusing target classes with each other or potentially misclassifying non-target classes as target ones, resulting in decreased classification performance; (3) Neglecting fine-tuning of the backbone network: Typically, the backbone network only participates in training during the base class phase, and not in the fine-tuning stage, preventing the model from fully utilizing novel class samples.

Based on these consideration, we propose a Faster RCNN model based on Smooth variational autoencoder and Enhanced Detection Head (S-EDH-Faster RCNN). It adopts a two-stage training process of involving base training and fine-tuning. In the base training stage, a smooth variational autoencoder learns realistic feature distributions from extensive training data. In the fine-tuning stage, the backbone network is adaptively fine-tuned to capture and learn the characteristics of novel class samples. Simultaneously, the smooth cariational autoencoder generates feature representations of novel class samples to increase the diversity of novel classes. Finally, the enhanced detection head computes the similarity scores between the enhanced RoI features and the feature prototypes, calibrating classification scores with the similarity scores to improve classification accuracy.

In summary, the main contributions of our work are:

1. We propose a Faster RCNN model based on smooth variational autoencoder and enhanced detection head for few-shot pipeline DR defect detection. It reconstructs the feature distribution via the smooth variational autoencoder and improves defect classification scores with an enhanced detection head. During fine-tuning, the backbone network is adaptively fine-tuned to learn the features of novel class samples.
2. We present a sample feature distribution generation method based on the smooth variational autoencoder (S-VAE), enriching the feature representation of data by generating diversified novel class sample features.
3. We construct an enhanced detection head (EDH). It leverages a convolutional block attention-based center point classification calibration module to calculate feature prototypes and correct classification scores, thus enhancing the model's classification capabilities.
4. We design an adaptive fine-tuning strategy. It adaptively updates the primary convolution kernels of the backbone network during fine-tuning, allowing the model to better learn the characteristics of novel class samples.
5. The proposed method achieves superior results compared to the state-of-the-art models on both the self-made PIP-DET dataset and the publicly available NEU-DET dataset, demonstrating its effectiveness.

## 2 Related work

Our work is primarily related to three key tasks: variational autoencoders, detection heads, and fine-tuning strategies.

### 2.1 Variational autoencoder

Variational AutoEncoder (VAE), introduced by Kingma and Welling [26], focuses on probabilistic modeling of latent variables from input data to generate realistic new samples. VAEs utilize an encoder to convert input data into latent variables, followed by a decoder to reconstruct the data. During training, the loss function combines the reconstruction error with the KL divergence between the latent variable distribution and a standard normal distribution to optimize the model.

Despite VAE's ability to generate realistic new samples, they face main two challenges. First, the loss function combining reconstruction error and KL divergence might produce samples that are not sufficiently close to real ones [27]. Second, the standard VAE may struggle to handle complex data distributions, limiting their generative capacity and application scope [28]. To enhance VAE's performance, various improvement methods have been proposed, mainly focusing on reconstruction loss and regularization term enhancements [27, 29, 30].

The improvements based on reconstruction loss involve designing diverse reconstruction losses to encourage the VAE to generate more realistic samples or features. For example, Chen et al. proposed the Log Hyperbolic Cosine Variational

AutoEncoder (Log-cosh VAE) [31], which reduces the impact of outliers by using log-cosh loss. Naesseth et al. introduced the Gamma Variational AutoEncoder (Gamma-VAE) [32], which dynamically adjusts the value of learnable parameters to generate higher quality reconstructed features. Tomczak and Welling proposed the Variational Mixture of Posteriors Prior Variational AutoEncoder (VampPrior VAE) [33], which uses a flexible mixture prior to improve reconstruction capabilities. Wang et al. suggested the Conditional Variational AutoEncoder (CVAE) [34], which incorporates conditional information in the reconstruction loss to make generated samples correspond better to given conditions. Bai et al. presented the Hierarchical Variational AutoEncoder (HVAE) [35], which introduces hierarchical structures in the reconstruction loss to allow multi-scale representation learning of the data.

The enhancements based on regularization focus on optimizing the KL divergence to increase sample diversity. Maxwell et al. proposed $\beta$-Variational Auto-Encoder ($\beta$-VAE) [36], introducing a hyperparameter $\beta$ in the KL divergence term to generate higher-quality samples. Kumar et al. introduced the Disentangled Inferred Prior Variational Auto-Encoder (DIP-VAE) [37], which adds new terms to the loss function to disentangle the latent space, enhancing the independence of latent representations. Kolouri et al. proposed the Sliced-Wasserstein Autoencoder (SWAE) [38], incorporating the Sliced-Wasserstein distance in the loss function to minimize the distance between the model output distribution and the target distribution, thereby improving sample quality. Pineau et al. put forward the Categorical Variational AutoEncoder (CatVAE) [39], which introduces discrete latent spaces to enhance the model's interpretability.

Moreover, some researchers have integrated enhancements in both reconstruction loss and regularization. For instance, Cai et al. introduced the Two-Stage Variational AutoEncoder (Two-Stage VAE) [40], where the training process consists of two stages: the first stage focuses on optimizing the encoder, and the second stage aims to improve reconstruction quality and regularize the latent space, thus enhancing both reconstruction loss and regularization.

Although these VAEs and their improvements perform well with sufficient training data, they still encounter challenges in few-shot scenarios. On the one hand, noise and outliers can interfere with VAE's ability to capture actual data distribution, reducing feature quality. On the other hand, the limited distribution information obtained by VAEs may include random or incidental correlations among some features, leading to misjudging these features as interdependent. Therefore, we aim to improve both the reconstruction loss and regularization of the standard VAE to enhance its performance under few-shot conditions.

## 2.2 Detection head

The detection head of Faster RCNN comprises a classification branch and a localization regression branch [41]. In few-shot detection, the base class is the class that has sufficient labeled data in the training phase, through which the model learns generic features; while the novel class is the class that has scarce data in the training phase, and the model needs to rely on what it learns from the base class to detect the novel class effectively with only a small number of samples. The limited number of novel class samples makes it challenging for the model to adequately learn the characteristics of new targets, often resulting in classification errors [42, 43]. To address this issue, various improvement methods have been proposed, which can be categorized into two types based on the structure modification of the detection head: adding new branches and optimizing existing structures.

The former methods introduce one or more independent branches to enhance the performance of the detection head. For instance, Li et al. proposed a 'Disentangle and Remerge' method (DandR) [44], introducing an auxiliary branch for knowledge distillation in the detection head of Faster RCNN to reduce model parameters. Sun et al. proposed the Few-Shot Object Detection via Contrastive Proposal Encoding (FSCE) [45], adding a contrastive branch in the detection head to decrease intra-class variance and increase inter-class differences, thus reducing classification errors under few-shot conditions.

The later methods focus on enhancing the performance of current functional modules, without adding new independent branches. For example, Yang et al. introduced the Context-Transformer (CT) [46], incorporating context information from images into the object classifier to strengthen feature representation. Qiao et al. proposed the Prototypical Calibration Block (PCB) [20], which calculates similarity scores between feature prototypes and RoI features to calibrate the softmax scores of the classification branch, thereby improving classification accuracy.

While these improved detection heads enhance classification performance to some extent, they also have limitations. Adding new branches can introduce independent functional modules to improve detection head performance, but when training samples are limited, the model either struggles to distinguish similar category features accurately or confuses novel classes with background classes.

## 2.3 Fine-tuning Strategy

In fine-tuning-based few-shot object detection, selectively freezing certain network layers while tuning specific parameters prevents the model from forgetting knowledge acquired

during base training and avoids overfitting on novel class samples [47]. To balance performance on base and novel classes, researchers have proposed various fine-tuning strategies, classified into two categories based on network layer updating strategies: freezing all components except the detection head and unfreezing some components beyond the detection head.

Strategies that freeze all components except the detection head maintain the backbone network and region proposal network (RPN) in a frozen state, updating only specific layers in the detection head. For example, Wang et al. proposed the Two-stage Fine-tuning Approach (TFA) [8], which freezes all components except the detection head during the fine-tuning stage to prevent degradation in base class detection performance. Mpampis et al. introduced Symmetric Fine-Tuning [48], which freezes the backbone network and RPN but fine-tunes the first and last few layers of the detection head to alleviate accuracy constraints and overfitting compared to traditional methods that only fine-tune the final layers of the detection head.

Strategies that unfreeze some components beyond the detection head aim to balance detection performance between base and novel classes, by unfreezing limited layers of the backbone network and RPN. Fan et al. proposed the Bias-Balanced RPN [49], updating the classification layer of the RPN and the last layer of the classification and bounding box regression branches in the detection head during fine-tuning. Hao et al. introduced Few-shot object detection via online inferential calibration (FSOIC) [50], employing a hierarchical freezing strategy to selectively unfreeze different network modules based on the number of training samples (shot count). Specifically, when samples are scarce, the backbone network is frozen, and only the classifier and regressor are updated; as the number of samples increases, Attention-FPN, RPN, and RoI are gradually unfrozen to better adapt to learning novel class features.

Although these fine-tuning methods achieve some success in enhancing model adaptability, completely freezing the backbone network during fine-tuning limits the model's feature extraction capabilities to knowledge acquired during base class training, underutilizing novel class features. Additionally, static freezing strategies lack flexibility, failing to dynamically choose network layers for updating based on data complexity.

## 3 Method

This section will first outline the overall architecture of the S-EDH-Faster RCNN, then detail its core components: Smooth Variational Autoencoder (S-VAE), Enhanced Detection Head (EDH), and Adaptive fine-tuning strategy. Finally, the loss functions used to train this model will be given.

### 3.1 Overall architecture

To enhance the accuracy of defect detection in pipeline DR images with few samples, we propose a Faster RCNN based on Smooth variational autoencoder and Enhanced Detection Head, referred to as S-EDH-Faster RCNN. This model optimizes Faster RCNN through three avenues: generating reconstructed features using a Smooth Variational Autoencoder (S-VAE), improving classification precision with an enhanced detection head, and dynamically adjusting the backbone network via an adaptive fine-tuning strategy. Figure 1 illustrates the overall structure of this model.

As shown in Fig. 1, due to the scarcity of novel class samples, the model uses a smooth variational autoencoder to learn the latent distribution of input data during feature extraction, thereby generating new training features. To enhance the classification accuracy between novel and background classes, we incorporate Center Point Classification

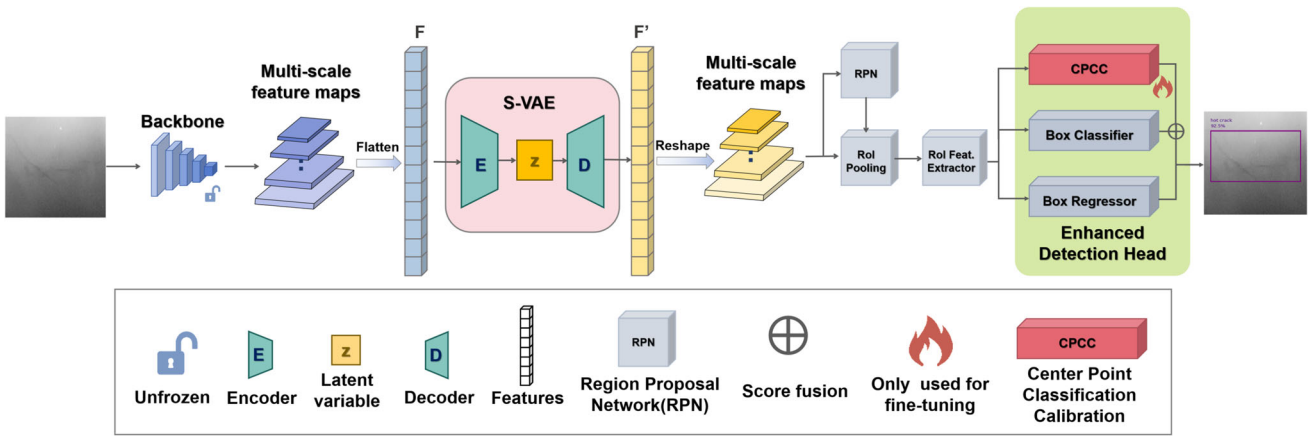



**Fig. 1** The overall architecture of the S-EDH-Faster RCNN model. It combines a training strategy of a Smooth Variational Autoencoder (S-VAE), an enhanced detection head, and an adaptive fine-tuning backbone network to optimize defect detection performance on few-shot data

Calibration (CPCC) into the classifier for improved optimization. Finally, to enable the model to effectively learn novel class features, we synchronously adjusted the backbone network during the fine-tuning stage through an adaptive fine-tuning style.

The training process of S-EDH-Faster RCNN consists of two stages: base training and fine-tuning. In the base training stage, the model is trained using abundant base class sample data ($D_{train} = D_{base}$). For an input image, the multi-scale feature maps are first extracted using the backbone network. These multi-scale features are then flattened into vectors and fed into the Smooth Variational Autoencoder (S-VAE) to obtain reconstructed features $\boldsymbol{F}^{'}$. These reconstructed features are passed to the region proposal network (RPN), generating a series of candidate regions that, after RoI pooling and feature extraction, provide RoI features for subsequent detection. Finally, the detection head classifies and locates these RoI features, outputting defect categories and locations.

In the fine-tuning stage, the model is fine-tuned using a balanced dataset containing novel class samples and randomly sampled base class samples ($D_{train} = D_{novel} \cup D_{base'}$). There are two main differences with the base training stage: First, to enhance the model's performance in detecting novel class samples, the backbone network is not completely frozen but adaptively selects the first $k$ convolutional kernels for updating. Second, to increase the classification precision of novel features, we design an enhanced detection head. Before computing the class scores, the cosine similarity scores between the feature prototypes and the enhanced novel features are calculated and weighted with the softmax scores to improve classification accuracy. Specifically, for RoI features, ResNet18 is utilized to extract features highly relevant to classification. These features are then input into a convolutional block attention module (CBAM) to obtain the RoI features with enhanced classification characteristics. CBAM is a type of attention module that enhances model focus by prioritizing certain features. It specifically does this through channel attention, which focuses on the most informative features across channels, and spatial attention, which highlights important spatial regions in the input. This dual mechanism helps direct the model's attention to the most relevant features for better performance on tasks such as image recognition. The feature prototype is the most characteristic point within a specific class. Specifically, it is a summary of the features that best define a particular class. In classification tasks, this prototype can be used to enhance the model's interpret ability and improve its accuracy by comparing new data points with these representative feature prototypes. Finally, the cosine similarity scores between the enhanced RoI features and each class prototype feature are calculated and weighted with the classifier's softmax scores to form the final classification scores.

### 3.2 Smooth variational autoencoder

To make the VAE better deal with noise and learn independent features, we propose a Smooth Variational Autoencoder (S-VAE), utilizing Log-cosh and smooth L1 as reconstruction losses to handle large errors more effectively. Simultaneously, we use the distance between latent variables as a regularization term to decouple latent variables in the latent space.

Specifically, let $\boldsymbol{h}_i$ denote the feature of real data, $\hat{\boldsymbol{h}}_i$ represent the reconstructed features by S-VAE, $\boldsymbol{t}$ indicate the error between actual and predicted values $|\boldsymbol{h}_i - \hat{\boldsymbol{h}}_i|$, $z$ is the latent variable, $\phi$ represent the parameters of the encoder, $q_\phi(z)$ mean the distribution of latent variable $z$, and $p(z)$ indicate the prior distribution. Log-cosh and smooth L1 can be expressed as:

$$L_{\log-\cosh}(\boldsymbol{h}, \hat{\boldsymbol{h}}) = \frac{1}{a}\sum_i \log\left(\cosh(a(\boldsymbol{h}_i - \hat{\boldsymbol{h}}_i))\right)$$

$$= \frac{1}{a}\sum_i \log\left(\frac{e^{a\boldsymbol{t}} - e^{-a\boldsymbol{t}}}{2}\right) \to \begin{cases} |\boldsymbol{t}| - \frac{1}{a}\log 2, & \text{when } |\boldsymbol{t}| \to \infty \\ 0.5a\boldsymbol{t}^2, & \text{when } |\boldsymbol{t}| \to 0 \end{cases} \tag{1}$$

$$L_{\text{smooth L1}} = \begin{cases} 0.5\boldsymbol{t}^2, & \text{if } |\boldsymbol{t}| < 1 \\ |\boldsymbol{t}| - 0.5, & \text{otherwise} \end{cases} \tag{2}$$

where $a$ is a positive hyperparameter adjusting the sensitivity of the loss function to errors. When the error $|\boldsymbol{t}|$ is large, log-cosh and smooth L1 approximate the L1 norm, showing robustness to large errors. When $|\boldsymbol{t}|$ is small, they approximate the L2 norm, exhibiting smoothing advantages.

The improved reconstruction loss can be expressed as:

$$L_{\text{recon}}(|\boldsymbol{t}|) = \alpha L_{\text{log-cosh}}(|\boldsymbol{t}|) + (1-\alpha)L_{\text{smooth L1}}(|\boldsymbol{t}|) \tag{3}$$

where $\alpha$ is a hyperparameter between 0 and 1 controlling the contributions of the two losses.

Additionally, considering that each dimension in the latent space corresponds to a latent variable, these variables ideally should independently represent different generative factors. If there are correlations among latent variables, they may carry redundant information, affecting the model's expressiveness by failing to independently represent different generative factors. Therefore, we use the distance between latent variables as a regularization term to achieve decoupling, which is represented as:

$$D(q_\phi(z)||p(z)) = \lambda_{od}\sum_{i\neq j}\left[\text{Cov}_{q_\phi(z)}[z]\right]_{ij}^2 + \lambda_d\sum_i\left(\left[\text{Cov}_{q_\phi(z)}[z]\right]_{ii} - 1\right)^2 \tag{4}$$

where $\sum_{i\neq j}\left[\text{Cov}_{q_\phi(z)}[z]\right]_{ij}^2$ represents the sum of the off-diagonal elements of the covariance matrix, penalizing

non-diagonal elements, and $\sum_i \left( \left[ \mathrm{Cov}_{q_\phi(z)}[z] \right]_{ii} - 1 \right)^2$ represents the sum of the squared differences between the diagonal elements of the covariance matrix and 1, adjusting the covariance matrix of the latent variables to ensure that the variance of each latent variable approximates 1, maintaining an appropriate level of decoupling in the latent space, $\lambda_{od}$ and $\lambda_d$ are hyperparameters.

In summary, the total loss of the improved S-VAE aims to incorporate the above factors and is expressed as:

$$L_{\mathrm{S-VAE}} = L_{\mathrm{recon}}(|\boldsymbol{t}|) + D_{KL}(q_{\phi(z|\boldsymbol{x})}||p(z)) + \beta D(q_\phi(z)||p(z)) \quad (5)$$

where $L_{\mathrm{recon}}(|\boldsymbol{t}|)$ denotes the reconstruction loss calculated using log-cosh and smooth L1 losses, $D_{KL}(q_\phi(z|\boldsymbol{x})||p(z))$ represents the KL divergence between the encoder output $q_{\phi(z|\boldsymbol{x})}$ and the prior distribution $p(z)$, $D(q_\phi(z)||p(z))$ signifies the decoupling regularization term, and $\beta$ is the weight of the term $D(q_\phi(z)||p(z))$.

## 3.3 Enhanced detection head

In small-sample defect detection tasks, low-quality scores generated by the model often lead to two issues: high-score false positives and low-score missed detections. That is, the model erroneously identifies non-target class samples (negative samples) as target class samples (positive samples) or fails to correctly recognize target class samples (positive samples) [51–53]. To address this challenge, we present an enhanced detection head that utilizes the similarity scores between feature prototypes and sample features to calibrate classification scores, thereby reducing high-score false positives and mitigating low-score missed detections. Figure 2 shows the overall structure of the enhanced detection head.

Specifically, it starts by leveraging a shallow network, ResNet18, to further extract features highly pertinent to classification. These classification-related features are then fed into a CBAM to obtain the enhanced RoI features. Finally, the cosine similarity between these enhanced RoI features and the prototype features of each class is computed, and this similarity score is fused with the softmax scores generated by the classifier to produce the final classification scores.

### 3.3.1 Convolutional block attention module

The Convolutional Block Attention Module (CBAM) calculates the weights of features in channel attention and spatial attention modules separately, enhancing key features related to the classification task while suppressing irrelevant features. Figure 3 illustrates the structure of CBAM.

As depicted in Fig. 3, CBAM mainly comprises two parts: channel attention and spatial attention. Channel attention computes the importance weight of each feature channel, strengthening significant channels; spatial attention calculates

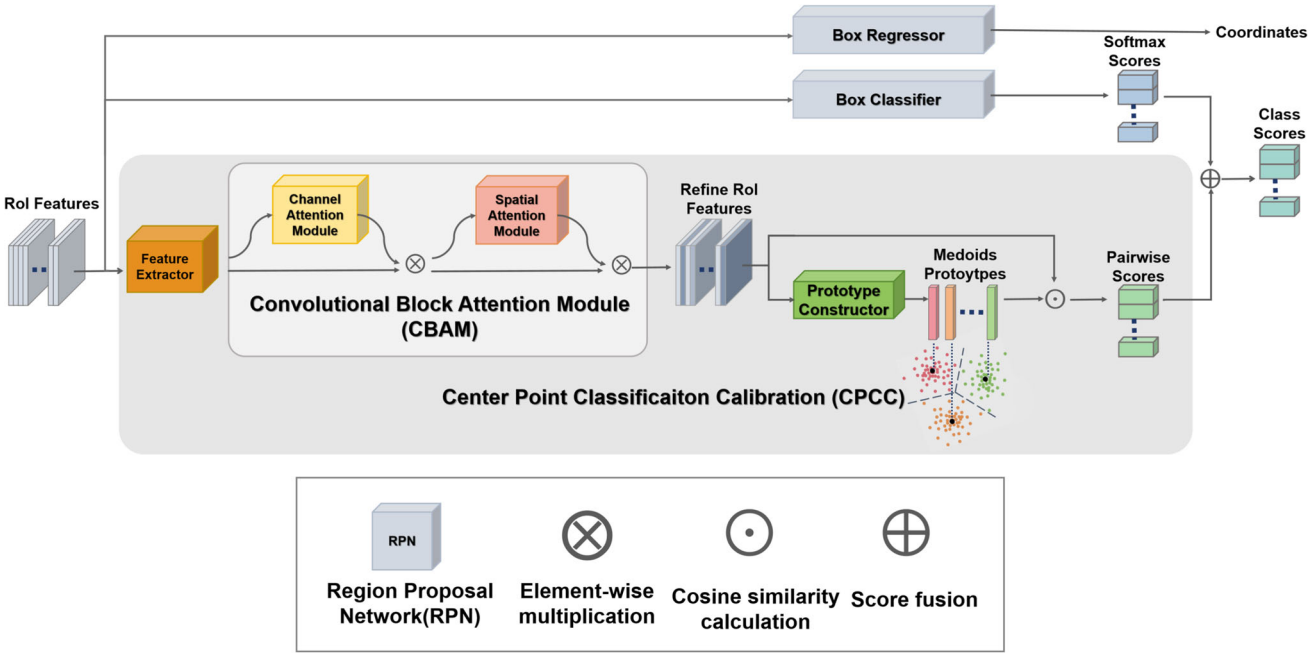



**Fig. 2** The structure of the enhanced detection head. The enhanced detection head uses a feature extractor and a Convolutional Block Attention Module (CBAM) to further refine the features, which are then paired with a Centroid Classification Calibration (CPCC) module to improve classification accuracy through score fusion

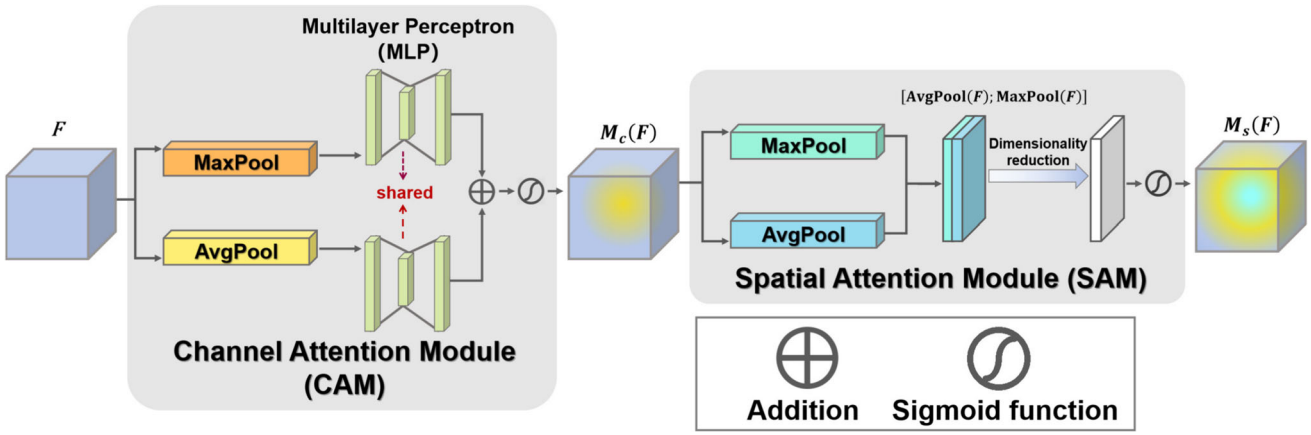



**Fig. 3** The structure of the convolutional block attention module(CBAM). CBAM uses the Channel Attention Module (CAM) and the Spatial Attention Module (SAM) to enhance feature expression in collaboration

the importance weight of each spatial position in the feature map, highlighting crucial areas.

In the channel attention calculation phase, average pooling and max pooling are applied to the feature vector, which is then input into a Multi-Layer Perceptron (MLP). The outputs are summed and processed by the sigmoid function to get the final channel attention map. Let $F$ be the classification-relevant features extracted by ResNet18, $C$ represents the number of feature channels, AvgPool denotes the global average pooling operation, MaxPool denotes the global maximum pooling operation, and $\sigma(\cdot)$ is the sigmoid function. The channel attention process can be expressed as:

$$M_c(\boldsymbol{F}) = \sigma\left(\mathrm{MLP}(\mathrm{AvgPool}(\boldsymbol{F})) + \mathrm{MLP}(\mathrm{MaxPool}(\boldsymbol{F}))\right) \tag{6}$$

In the spatial attention calculation phase, average pooling and max pooling are first applied to the feature map along the channel dimension. The two results are concatenated and processed by a $7 \times 7$ convolution layer, followed by a sigmoid function to obtain the spatial attention map. The spatial attention process can be expressed as:

$$M_s(\boldsymbol{F}) = \sigma\left(Conv^{7\times7}\left([\mathrm{AvgPool}(\boldsymbol{F}); \mathrm{MaxPool}(\boldsymbol{F})]\right)\right) \tag{7}$$

where $Conv^{7\times7}$ represents a $7 \times 7$ convolution layer, and $[\mathrm{AvgPool}(\boldsymbol{F}); \mathrm{MaxPool}(\boldsymbol{F})]$ denotes the concatenation of average pooling and max pooling results along the channel dimension, resulting in a $2 \times H \times W$ feature map.

### 3.3.2 Center Point classification calibration

Traditional feature prototypes are often generated by averaging the feature vectors of all samples in a class. However, it is susceptible to outliers, making the feature prototype inaccurately represent the actual feature distribution of the class. Especially in small-sample scenarios, when the sample size is limited or intra-class feature distribution is dispersed, the prototype computed by averaging may deviate from the actual class center, affecting classification performance. Hence, we adopt the Medoids method to compute feature prototypes. By selecting the feature with the minimum distance to other features in the feature space as the prototype, it better represents the class center and reduces the impact of outliers.

Specifically, let there be a feature set $\{\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_n\}$, using Euclidean distance to define a distance metric $d(\boldsymbol{F}_i, \boldsymbol{F}_j)$:

$$d(\boldsymbol{F}_i, \boldsymbol{F}_j) = \sqrt{\sum_{k=1}^{m} (\boldsymbol{F}_{ik} - \boldsymbol{F}_{jk})^2} \tag{8}$$

where $\boldsymbol{F}_{ik}$ and $\boldsymbol{F}_{jk}$ represent the values of features $\boldsymbol{F}_i$ and $\boldsymbol{F}_j$ in the $k$-th dimension, respectively.

For the $i$-th feature $\boldsymbol{F}_i$, we first calculate the sum of distances to all other features $S(\boldsymbol{F}_i)$:

$$S(\boldsymbol{F}_i) = \sum_{j=1}^{n} d(\boldsymbol{F}_i, \boldsymbol{F}_j) \tag{9}$$

Then, find the feature $\boldsymbol{F}_p$ that minimizes $S(\boldsymbol{F}_i)$:

$$\boldsymbol{F}_p = \arg\min_{\boldsymbol{F}_i} S(\boldsymbol{F}_i) \tag{10}$$

The feature prototype for each class is calculated from all the enhanced RoI feature vectors of that class, with $\boldsymbol{F}_p$ as the selected feature prototype.

After obtaining the feature prototype $\boldsymbol{F}_p$, it is used in the classification calibration module. The main task of this module is to select the enhanced novel class RoI features outputted by the RoI feature extractor and compare them with the

feature prototypes of each class, calculating the cosine similarity between them. The similarity score is then combined with the classification score from the classifier to yield the final classification score. During the testing phase, the classification calibration module performs score calibration for all classes' enhanced RoI features. For each class $c$, the feature prototype $\boldsymbol{F}_p$ can be computed according to the formula. After getting the feature prototype, the Enhanced Detection Head calculates the cosine similarity between the RoI feature vector $\boldsymbol{F}_{\text{RoI}}$ and each feature prototype vector $\boldsymbol{F}_p$. The cosine similarity can be expressed as:

$$\text{CosSim}(\boldsymbol{F}_{\text{RoI}}, \boldsymbol{F}_p) = \frac{\boldsymbol{F}_{\text{RoI}} \cdot \boldsymbol{F}_p}{\|\boldsymbol{F}_{\text{RoI}}\| \|\boldsymbol{F}_p\|} \tag{11}$$

where $\boldsymbol{F}_{\text{RoI}} \cdot \boldsymbol{F}_p$ represents the dot product of vectors, and $\|\boldsymbol{F}_{\text{RoI}}\|$ and $\|\boldsymbol{F}_p\|$ represent the norms of vectors $\boldsymbol{F}_{\text{RoI}}$ and $\boldsymbol{F}_p$, respectively.

Simultaneously, the RoI feature vector $\boldsymbol{F}_{\text{RoI}}$ is also input into the classifier to obtain the preliminary classification scores $\boldsymbol{S}_{\text{class}}$, where $\boldsymbol{S}_{\text{class}} = [s_1, s_2, \ldots, s_C]$, with $C$ representing the number of classes, and $s_i$ representing the score of the $i$-th class. The cosine similarity scores for each class are weighted and fused with the corresponding classification scores to get the final classification scores $\boldsymbol{S}_{\text{final},c}$:

$$\boldsymbol{S}_{\text{final},c} = \gamma \text{CosSim}(\boldsymbol{F}_{\text{RoI}}, \boldsymbol{F}_p) + (1 - \gamma)\boldsymbol{S}_{\text{final},c} \tag{12}$$

where $\gamma$ and $(1-\gamma)$ are the weight parameters used to balance the cosine similarity scores and softmax classification scores.

### 3.4 Adaptive fine-tuning

In the two-stage training process, traditional methods usually freeze the backbone network and only fine-tune the detection head. This strategy may result in the model failing to adequately learn the characteristics of novel class samples. To address this issue, we propose an adaptive fine-tuning method, which automatically selects and updates the top $k$ important convolutional kernels in the backbone network to ensure effective feature extraction from novel class samples.

The $L_2$ norm is utilized as a measure of the importance of convolutional kernels. The calculation formula is:

$$\|\boldsymbol{W}_i\|_2 = \sqrt{\sum_j \boldsymbol{w}_{ij}^2} \tag{13}$$

where $\boldsymbol{W}_i$ represents the $i$-th convolutional kernel, and $\boldsymbol{w}_{ij}$ represents the $j$-th parameter value within the $i$-th convolutional kernel.

After obtaining the $L_2$ norms of the convolutional kernels, these kernels are sorted by their importance, and the top $k$ important ones are selected for updating.

### 3.5 Loss functions

In the first stage, the RPN binary cross-entropy loss is used to optimize the model's ability to distinguish between foreground and background, which is represented as:

$$L_{RPN\_cls} = -\frac{1}{N}\frac{1}{N_{\text{anchor}}}\sum_{i=1}^{N}\sum_{j=1}^{N_{\text{anchor}}}\Big[\boldsymbol{y}_{ijc}\log(\boldsymbol{y}'_{ijc}) + (1-\boldsymbol{y}_{ijc})\log(1-\boldsymbol{y}'_{ijc})\Big] \tag{14}$$

where $\boldsymbol{y}_{ijc}$ represents whether the $j$-th anchor in the $i$-th sample is foreground ($y_{ijc} = 1$) or background ($\boldsymbol{y}_{ijc} = 0$), $\boldsymbol{y}'_{ijc}$ is the predicted foreground probability for the $j$-th anchor, $N$ is the number of training samples, and $N_{\text{anchor}}$ denotes the number of anchors per sample.

The regression loss in the RPN is used to optimize the model's accuracy in predicting the positions of bounding boxes, which is:

$$L_{RPN\_reg} = \frac{1}{N}\frac{1}{N_{\text{anchor}}}\sum_{i=1}^{N}\sum_{j=1}^{N_{\text{anchor}}}\text{smooth}_{L1}\left(\boldsymbol{y}'_{ijb} - \boldsymbol{y}_{ijb}\right)\cdot\boldsymbol{y}_{ijc} \tag{15}$$

where $\boldsymbol{y}_{ijb}$ represents the ground truth bounding box parameters for the $j$-th anchor in the $i$-th sample, and $\boldsymbol{y}'_{ijb}$ represents the predicted bounding box parameters, $N$ is the number of training samples, and $N_{anchor}$ is the number of anchor samples.

The bounding box classification loss optivmizes the model's ability to differentiate between different categories. It can be described as:

$$L_{cls} = -\frac{1}{N}\left[\sum_{i=1}^{N}\sum_{c=1}^{C}\boldsymbol{y}_{ic}\log(\boldsymbol{y}^*_{ic})\right] \tag{16}$$

where $\boldsymbol{y}_{ic}$ represents the true class label of the $i$-th sample, $\boldsymbol{y}^*_{ic}$ represents the predicted class probability, $C$ denotes the number of categories, and $N$ is the number of training samples.

The Smooth L1 loss for bounding box regression is used for precise localization of bounding boxes, which is:

$$L_{reg} = \frac{1}{N}\sum_{i=1}^{N}\text{smooth}_{L1}(\boldsymbol{y}^*_{ib} - \boldsymbol{y}_{ib}) \tag{17}$$

where $\boldsymbol{y}_{ib}$ represents the true bounding box parameters, $\boldsymbol{y}^*_{ib}$ represents the predicted bounding box parameters, and $N$ denotes the number of training samples.

The loss for S-VAE constrains the latent variables encoded to approximate a standard normal distribution. The formula

is:

$$L_{S-\mathrm{VAE}} = L_{recon} + D_{KL}(q_\phi(z|\boldsymbol{x})||p(z)) + \beta D(q_\phi(z)||p(z)) \tag{18}$$

where $L_{recon}$ represents the reconstruction loss calculated using log-cosh and smooth L1 losses, $D_{KL}(\boldsymbol{q}_{\phi(z|\boldsymbol{x})}||\boldsymbol{p}(z))$ represents the KL divergence between the encoder output $q_{(z|\boldsymbol{x})}$ and the prior distribution $p(z)$, and $D(q_{(z)}||p(z))$ denotes the decoupling regularization term, with $\beta$ being the weight parameter.

The total loss function during the base training stage can be expressed as:

$$L_{base} = L_{RPN\_cls} + L_{RPN\_reg} + L_{cls} + L_{reg} + L_{S-\mathrm{VAE}} \tag{19}$$

In the second stage, in addition to the losses from the first stage, an additional similarity loss for the EDH is introduced. The EDH generates the final classification score by computing the cosine similarity score between RoI features and feature prototypes, $\mathrm{CosSim}(\boldsymbol{F}_{\mathrm{RoI}}, \boldsymbol{F}_p)$, and fusing it with the softmax score. The loss is:

$$L_{similarity} = -\sum_{i=1}^{N}\sum_{c=1}^{C}\left(\mathrm{CosSim}(\boldsymbol{F}_{\mathrm{RoI}}, \boldsymbol{F}_p)\cdot \boldsymbol{y}_{ic}\right) \tag{20}$$

where $\mathrm{CosSim}(\boldsymbol{F}_{\mathrm{RoI}}, \boldsymbol{F}_p)$ represents the cosine similarity between the RoI feature vector $\boldsymbol{F}_{\mathrm{RoI}}$ and the feature prototype $\boldsymbol{F}_p$, and $\boldsymbol{y}_{ic}$ represents the true class label of the $i$-th sample, $N$ is the number of training samples, and $C$ is the counts of the classes.

The total loss function during the fine-tuning stage can be expressed as:

$$\begin{aligned} L_{fine\_tuning} = {} & L_{RPN\_cls} + L_{RPN\_reg} + L_{cls} + L_{reg} \\ & + L_{S-\mathrm{VAE}} + L_{similarity} \end{aligned} \tag{21}$$

## 4 Experimental Results

To validate the effectiveness of the proposed method, this section compares S-EDH-Faster RCNN with other models. It will introduce the datasets used, evaluation metrics, experimental details, and a detailed comparison with existing methods and ablation experiments, followed by the corresponding analysis. All experiments were conducted in a Linux environment using the PyTorch-based deep learning framework on a server equipped with an NVIDIA RTX A4000 GPU.

### 4.1 Dataset

To evaluate the performance of our method in few-shot object detection tasks, experiments were conducted on a self-made gas pipeline defect image dataset (PIP-DET). The PIP-DET dataset covers 20 defect categories, totaling 6,010 samples, with 4,258 samples in the training set and 1,752 samples in the test set. The images were annotated using LabelImg. For fair comparison, the dataset partitioning method follows previous works [8, 19, 54, 55]. Specifically, on PIP-DET dataset, 15 classes are randomly selected as base classes, while the remaining 5 classes are defined as novel classes. Each novel class contains K=1, 2, 3, 5, 10 annotated training samples to simulate few-shot scenarios. In this study, three random splits named split1, split2 and split3 were considered. Table 1 provides detailed information about PIP-DET, and Fig. 4 shows annotation examples of different defect images.

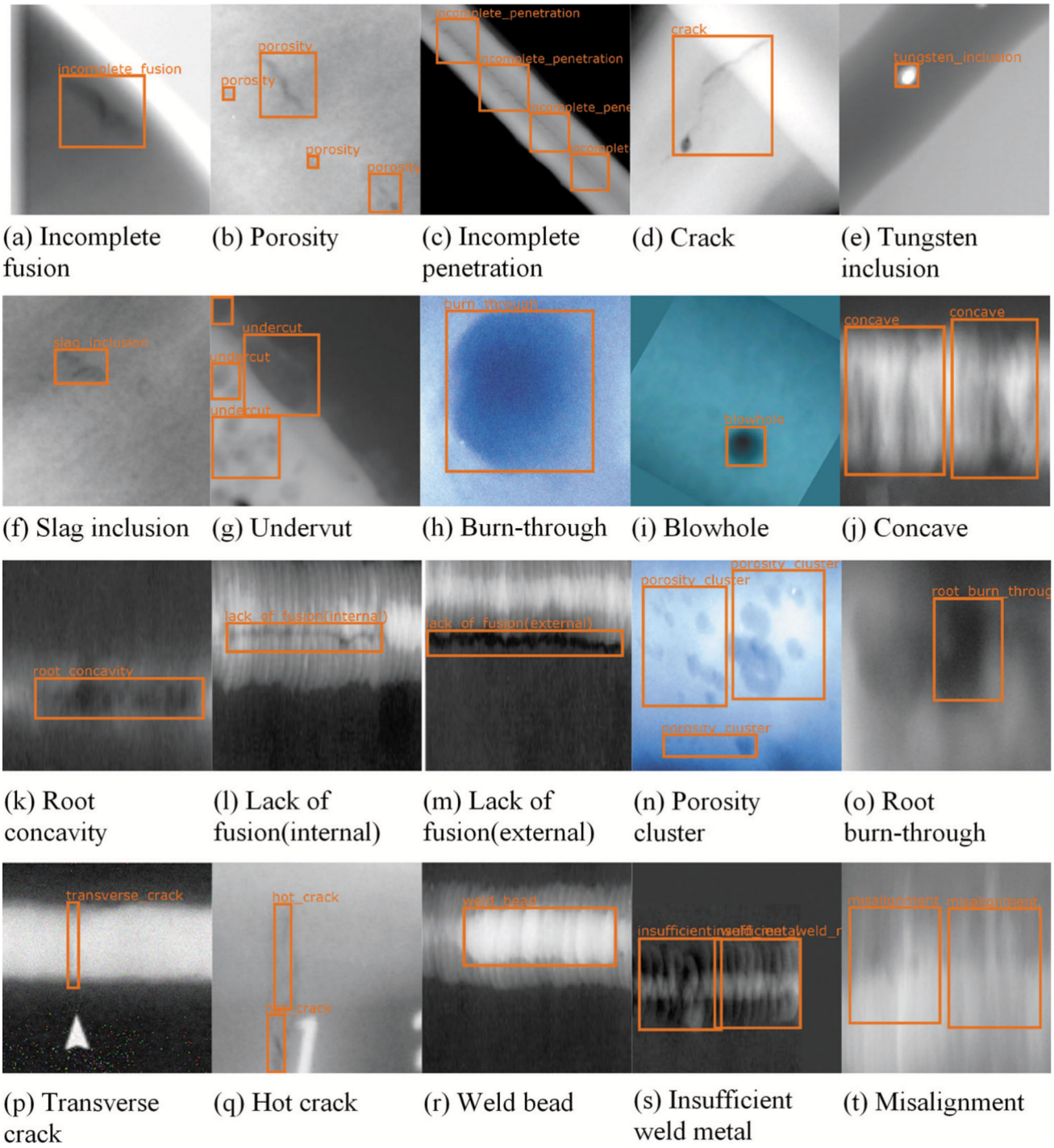
### 4.2 Evaluation metrics

Mean Average Precision (mAP) represents the average precision across all categories, calculated as:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{22}$$

**Table 1** The detail information of PIP-DET dataset

| Class | Name | Number | Class | Name | Number |
|---|---|---|---|---|---|
| 1 | Incomplete fusion | 694 | 11 | Root concavity | 417 |
| 2 | Porosity | 697 | 12 | Lack of fusion (internal) | 217 |
| 3 | Incomplete penetration | 699 | 13 | Lack of fusion (external) | 327 |
| 4 | Crack | 389 | 14 | Porosity cluster | 243 |
| 5 | Tungsten inclusion | 699 | 15 | Root burn-through | 385 |
| 6 | Slag inclusion | 710 | 16 | Transverse crack | 340 |
| 7 | Undercut | 286 | 17 | Hot crack | 271 |
| 8 | Burn-through | 364 | 18 | Weld bead | 482 |
| 9 | Blowhole | 417 | 19 | Insufficient weld metal | 300 |
| 10 | Concave | 283 | 20 | Misalignment | 392 |




(a) Incomplete fusion (b) Porosity (c) Incomplete penetration (d) Crack (e) Tungsten inclusion

(f) Slag inclusion (g) Undervut (h) Burn-through (i) Blowhole (j) Concave

(k) Root concavity (l) Lack of fusion(internal) (m) Lack of fusion(external) (n) Porosity cluster (o) Root burn-through

(p) Transverse crack (q) Hot crack (r) Weld bead (s) Insufficient weld metal (t) Misalignment

**Fig. 4** Example images of defects with annotations on PIP-DET dataset

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{23}$$

$$\text{AP}_i = \sum_{j=0}^{n-1} (R_{j+1} - R_j) p_j^* \tag{24}$$

$$\text{mAP} = \frac{1}{N_c} \sum_{i=1}^{N_c} \text{AP}_i \tag{25}$$

where precision and recall represent the accuracy of model predictions and the ability to identify positive samples; TP, FP, and FN denote the number of true positive, false positive, and false negative samples, respectively. $n$ denotes the number of interpolations performed on the precision-recall curve, and the recall rate after interpolation is: $R_j = \frac{j}{n}, \quad j \in \{0, 1, 2, \ldots, n\}, \quad p_j^*$ is the precision at $R_j$. $AP_i$ represents

the average precision for the $i$-th class; $N_c$ represents the number of categories.

Based on mAP, different average precision metrics are used to measure model performance, including:

1. $mAP_{50}$: Average precision of all categories when the IoU threshold is 0.5.
2. bAP: Average precision of base classes when the IoU threshold is 0.5.
3. nAP: Average precision of novel classes when the IoU threshold is 0.5.

## 4.3 Implementation details

### 4.3.1 Parameter settings

ResNet-101 pretrained on ImageNet is adopted as the backbone network, with a mini-batch size of 16, optimized using Stochastic Gradient Descent (SGD). During the base training stage, the initial learning rate was set to $1 \times 10^{-3}$, with a momentum coefficient of 0.9 and a weight decay parameter of 1e-4. During the few-shot fine-tuning stage, the learning rate was adjusted to 0.0004.

### 4.3.2 Selection of S-VAE loss balancing coefficients $\alpha$ and $\beta$

In (3) and (5), the S-VAE loss function includes two adjustable parameters, $\alpha$ and $\beta$. $\alpha = \{0.1, 0.2, 0.3, 0.4, 0.5\}$ and $\beta = \{0.2, 0.3, 0.4, 0.5, 0.6\}$ are taken as candidate values. Using the model without S-VAE as a baseline, it experimented by gradually changing the values of $\alpha$ and $\beta$. Figure 5 shows the impact of parameters $\alpha$ and $\beta$ on the experimental results. Through heatmap analysis, it can be found that when $\alpha = 0.2$ and $\beta = 0.4$, the model achieved the best average precision, improving by 2.36% compared to the model without S-VAE. This indicates that the balance effect in the loss function is optimal under this parameter combination, maximizing model performance.

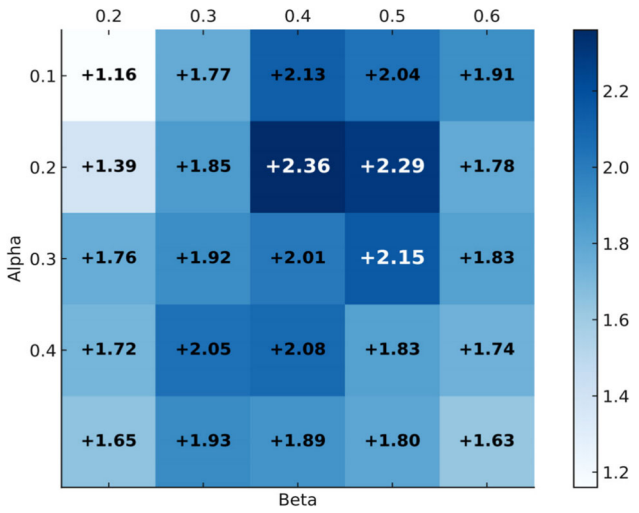



**Fig. 5** Experimental results of $\alpha$ and $\beta$ parameter comparison. The effects of different $\alpha$ and $\beta$ combinations on the model accuracy were compared in the form of heat maps

**Table 2** Results for different $\gamma$ values

| $\gamma$ | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 |
|---|---|---|---|---|---|---|
| mAP | 65.39 | 65.74 | 67.08 | **68.04** | 67.22 | 66.83 |

### 4.3.3 Selection of the enhanced detection head score balancing coefficient $\gamma$

As shown in (12), the final class score computed by the Enhanced Detection Head includes a balancing coefficient $\gamma$. It experimented with $\gamma = \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$ to compare the effects of different values. Table 2 presents the impact of different $\gamma$ values on the experimental results. From Table 2, it can be seen that, when $\gamma = 0.5$, the model achieved the highest detection result at 68.04%. Therefore, $\gamma = 0.5$ is selected as the final parameter.

### 4.3.4 Selection of fine-tuning update ratio $k$

During fine-tuning, different update ratios $k$ affect the number of updated convolution kernels, thus influencing the experimental results. Table 3 presents the impact of different update ratios on the experimental results, where the bold number denotes the best result and the underlined is the second best result. As shown in Table 3, when 26% of the convolutional kernels in the backbone network were updated per epoch, the model achieved the best average precision

**Table 3** Results of different $k$

| $k$ (%) | mAP | bAP | nAP |
|---|---|---|---|
| 0 | 65.04 | 70.10 | **49.84** |
| 20 | 65.96 | 73.57 | 43.12 |
| 21 | 65.93 | 75.05 | 38.55 |
| 22 | 66.82 | 75.12 | 41.93 |
| 23 | 66.84 | 74.60 | 43.55 |
| 24 | 66.29 | 75.58 | 38.42 |
| 25 | <u>67.82</u> | <u>76.04</u> | 43.14 |
| 26 | **68.07** | **76.07** | <u>44.08</u> |
| 27 | 67.24 | 75.75 | 41.70 |
| 28 | 67.70 | 74.63 | 43.94 |
| 29 | 66.01 | 74.39 | 40.87 |
| 30 | 65.96 | 73.57 | 43.12 |

**Table 4** Experimental results (nAP) of different methods on PIP-DET dataset

| Method | Novel Split1 | | | | | Novel Split2 | | | | | Novel Split3 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 5 | 10 | 1 | 2 | 3 | 5 | 10 | 1 | 2 | 3 | 5 | 10 |
| CME [23] | 23.43 | 30.86 | 35.28 | 38.91 | 41.30 | 19.52 | 21.14 | 33.44 | 34.39 | 37.82 | 17.08 | 21.19 | 31.63 | 32.52 | 40.94 |
| FSCE [45] | 28.96 | 34.68 | 38.57 | 44.18 | 51.13 | 20.32 | 23.58 | 31.11 | 30.01 | 33.28 | 20.13 | 24.28 | 32.58 | 39.29 | 39.64 |
| TFA [8] | 23.76 | 28.91 | 36.83 | 43.32 | 49.40 | 17.03 | 19.23 | 24.28 | 33.89 | 42.62 | 20.81 | 29.15 | 33.47 | 41.45 | 49.66 |
| Meta Faster RCNN [18] | 8.11 | 15.44 | 17.03 | 18.02 | 22.52 | 6.44 | 9.38 | 9.80 | 12.32 | 15.92 | 8.41 | 12.48 | 14.32 | 15.67 | 21.11 |
| DandR [44] | 31.90 | 35.08 | 39.08 | 43.54 | 50.65 | 29.03 | 30.86 | 35.25 | 40.97 | 46.50 | 21.67 | 28.22 | 35.22 | 39.48 | 42.04 |
| VFA [19] | 23.44 | 32.03 | 34.61 | 38.67 | 43.79 | 20.71 | 29.85 | 35.08 | 38.78 | 43.28 | 27.84 | 31.22 | 37.36 | 42.27 | 46.26 |
| DeFRCN [20] | 30.72 | 39.00 | 43.87 | 46.59 | 50.73 | 28.12 | 31.35 | 37.99 | 41.64 | 47.46 | **28.12** | 33.72 | 36.13 | 43.40 | 50.10 |
| CT [46] | 6.70 | 8.72 | 14.16 | 15.01 | 22.50 | 6.94 | 10.85 | 15.13 | 17.76 | 33.08 | 6.21 | 10.69 | 14.83 | 16.76 | 27.92 |
| RegAD [59] | 21.31 | 30.68 | 34.44 | 39.53 | 45.79 | 17.33 | 21.22 | 23.72 | 32.47 | 39.31 | 18.95 | 20.76 | 27.99 | 32.42 | 41.66 |
| Ours | **34.13** | **39.08** | **45.81** | **49.42** | **52.85** | **34.52** | **40.04** | **40.41** | **44.13** | **51.62** | 27.98 | **37.01** | **39.56** | **45.29** | **53.46** |

for all categories and base classes, with 68.07% and 76.07%, respectively. Therefore, the adaptive fine-tuning ratio was set to 26%.

### 4.4 Comparison with the state-of-the-art methods

To validate the effectiveness of S-EDH-Faster RCNN, experiments comparing it with other methods on PIP-DET dataset were conducted. Table 4 displays the mean Average Precision across three novel class splits (split1, split2, and split3), covering different sample shots (1, 2, 3, 5, 10 shots). In Table 4, bold numbers indicate the best results, while underlined numbers indicate the second-best results. Figure 6 shows the experimental results in a line chart, and Fig. 7 illustrates examples of detection results.

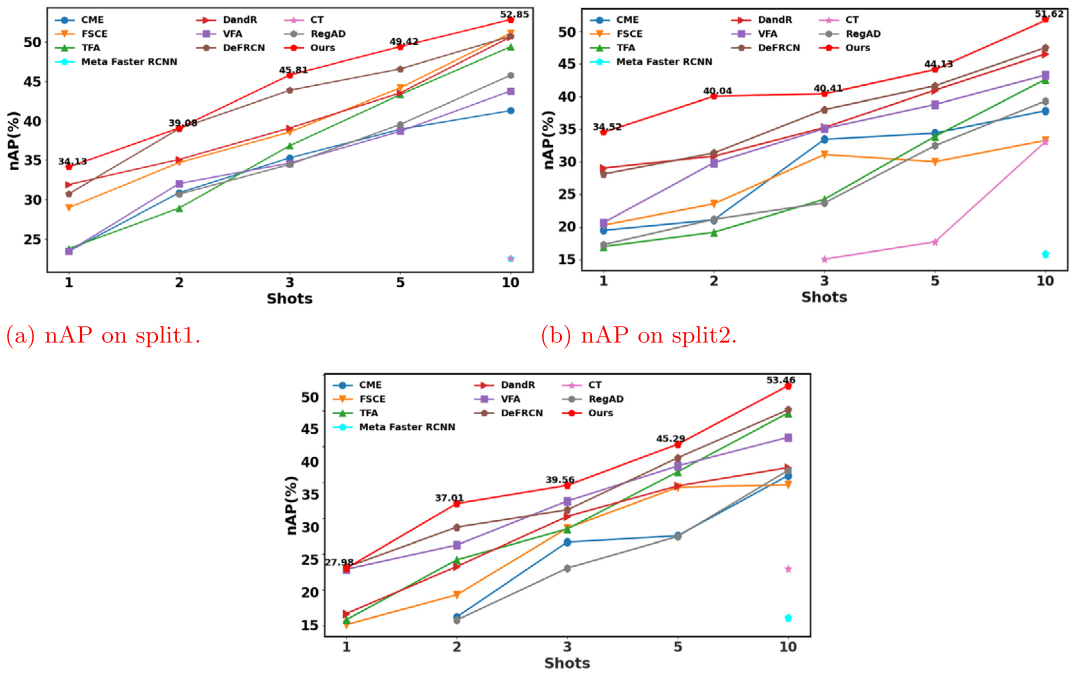
From the results in Table 4, Figs. 6 and 7, it can be drawn that: For split1, S-EDH-Faster RCNN achieved better detection results in most sample sizes. Specifically, it reached novel class detection accuracies of 34.13%, 39.08%, 45.81%, 49.42%, and 52.85% for 1, 2, 3, 5 and 10 shots, respectively. Compared to the second-best method, S-EDH-Faster RCNN improved by 2.23%, 0.08%, 1.94%, 2.83%, and 2.12%. Additionally, for split2, S-EDH-Faster RCNN consistently achieved the best detection accuracy for all sample sizes. For 1, 2, 3, 5 and 10 shots, it attained novel class average precisions of 34.52%, 40.04%, 40.41%, 44.13%, and 51.62%, surpassing the second-best method by 5.49%, 8.69%, 2.42%, 2.49%, and 4.16%. Besides, for split3, except for 1 shot, S-EDH-Faster RCNN achieved the best results in almost all scenarios. In detail, for 2, 3, 5 and 10 shots, it achieved 37.01%, 39.56%, 45.29%, and 53.46% novel class average precision, surpassing the second-best method by 3.29%, 2.20%, 1.89%, and 3.36%. For 1 shot, it obtained an average precision of 35.48%, just 0.14% lower than the top result. Overall, the proposed method not only outperformed other methods in most cases but also successfully identified and located novel class defects, further demonstrating its effectiveness.

### 4.5 Ablation study

This section evaluates the impact of different modules on the performance of the proposed method through a series of ablation experiments. Specifically, these experiments include testing S-VAE, enhanced detection head, and fine-tuning strategy, comparing them with other implementations, and using split1 with 10 shots as an example for analysis.

#### 4.5.1 Impact of different components on detection results

To explore the specific impact of each improved component on detection results, this section conducts experiments using different combinations of improvements, thereby determining the detection performance of S-EDH-Faster RCNN under




(a) nAP on split1.

(b) nAP on split2.

(c) nAP on split3.

**Fig. 6** Line graph of experimental results (nAP) on PIP-DET dataset. The average class precision (nAP) of S-EDH-Faster RCNN was compared with that of other advanced methods by using novel classes with different shots on three data splits.

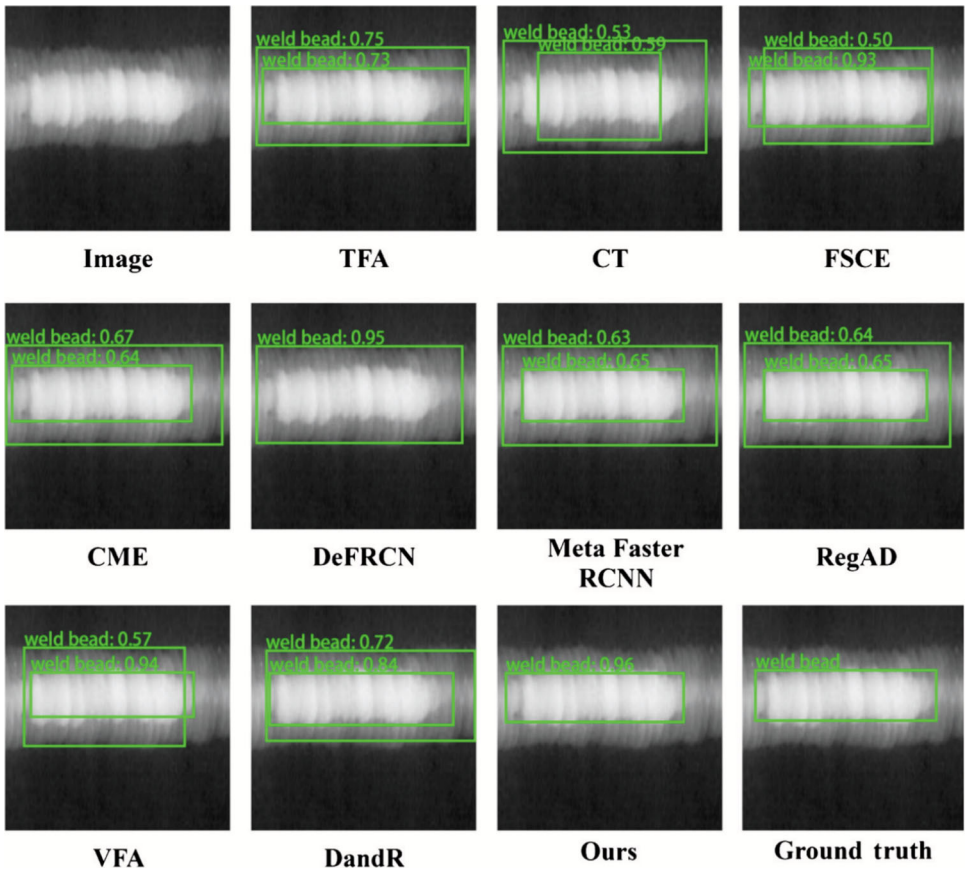



**Fig. 7** Detection examples of different methods on PIP-DET split1 (10 shots)

different enhancements. Table 5 presents the corresponding results.

From Table 5, it is clear that all different improvement combinations outperform the baseline Faster RCNN. When using one improvement method, when only using S-VAE, the improved model achieves an average precision of 65.00%; when only adopting the enhanced detection head, the average precision is 64.74%; when using adaptive fine-tuning alone, the average precision is 64.92%. When using two improvement methods, the combination of S-VAE and EDH achieves an average precision of 66.79%; the combination of S-VAE and adaptive fine-tuning achieves an average precision of 67.24%; the combination of enhanced detection head and adaptive fine-tuning achieves an average precision of 66.20%. When all three improvement methods are used simultaneously, the model's average precision increases to 68.07%. This indicates that S-VAE, EDH, and adaptive fine-tuning all effectively improve the model's detection performance, with their combined effect being the best, significantly enhancing the overall performance of S-EDH-Faster RCNN.

**Table 5** Ablation study of proposed modules

| Method | S-VAE | EDH | Adaptive | mAP Adjustment |
|---|---|---|---|---|
| Faster RCNN | | | | 62.64 |
| (i) | ✓ | | | 65.00 |
| (ii) | | ✓ | | 64.74 |
| (iii) | | | ✓ | 64.92 |
| (iv) | ✓ | ✓ | | 66.79 |
| (v) | ✓ | | ✓ | 67.24 |
| (vi) | | ✓ | ✓ | 66.20 |
| (vii) | ✓ | ✓ | ✓ | **68.07** |

#### 4.5.2 Impact of different variational autoencoders on detection results

To investigate the impact of different VAEs on detection results, experiments were conducted using various VAEs, and the detection performance of S-EDH-Faster RCNN under different conditions were assessed. Table 6 details the experimental results using different VAEs on the split1 with 10 shots scenario. S-VAE w/ Log-cosh, S-VAE w/ smooth L1, and S-VAE w/ DIP represent S-VAE solely using Log-cosh, smooth L1, and DIP loss functions, respectively. Figure 8 visualizes the feature space of different VAEs using t-SNE.

From Table 6 and Fig. 8, it can be got that: Compared with standard VAE and variants, S-VAE and its variants show

**Table 6** Performance comparison of different VAE

| Method | mAP | bAP | nAP |
|---|---|---|---|
| Standard VAE [26] | 64.17 | 70.00 | 46.66 |
| CVAE [34] | 66.05 | 70.42 | **52.92** |
| CatVAE [39] | 64.77 | 70.76 | 46.79 |
| Gamma-VAE [32] | 65.44 | 70.93 | 48.95 |
| $\beta$-VAE [36] | 63.37 | 68.60 | 47.67 |
| VampPriorVAE [33] | 65.56 | 70.29 | 51.36 |
| DIP-VAE [37] | 66.11 | 71.80 | 49.04 |
| SWAE [38] | 66.29 | 70.99 | 52.17 |
| Two-Stage VAE [40] | 63.36 | 70.58 | 41.68 |
| Log-cosh VAE [31] | 66.41 | 71.10 | 52.32 |
| HVAE [35] | 64.88 | 71.37 | 45.39 |
| S-VAE w/ Log-cosh | 66.27 | 70.83 | 52.58 |
| S-VAE w/ smooth L1 | 67.03 | 71.96 | 52.25 |
| S-VAE w/ DIP | 65.92 | 71.24 | 49.95 |
| S-VAE (Ours) | **68.07** | **73.14** | 52.85 |

improvements in mAP and bAP metrics. Specifically, S-VAE achieves the best overall detection performance, with an mAP of 68.07%, bAP of 73.14%, and nAP of 52.85%. Compared to other VAEs, S-VAE improves mAP by 1.66%-4.71% and bAP by 1.34%-4.54%. When S-VAE uses Log-cosh, smooth L1, and DIP loss functions simultaneously, it achieves optimal results, with mAP improved by 1.04%-2.15%, bAP by 1.18%-2.31%, and nAP by 0.27%-2.90% compared to S-VAE using other loss functions. This is because Log-cosh and smooth L1 as reconstruction losses handle large errors more effectively; additionally, using the distance between latent variables as a regularization term better decouples latent variables in the variational autoencoder's latent space.

Meanwhile, regarding feature distribution, the class distributions in the feature space generated by S-VAE are more separated, with clear class boundaries and higher intra-class compactness, superior to other methods. Specifically, categories in standard VAE, CatVAE, $\beta$-VAE, and SWAE are dispersed, with loose intra-class feature distributions, leading to blurred boundaries for some class feature clusters, making it difficult to accurately distinguish different clusters. CVAE, Gamma-VAE, VampVAE, and DIP VAE show tighter intra-class feature distributions, but their inter-class distances are too small, resulting in significant overlaps. Log-cosh VAE and HVAE show very tight class distributions with few overlaps. This demonstrates that S-VAE can generate high-quality samples, providing more valuable data for model training.

Overall, the various loss functions in S-VAE significantly enhance the model's detection performance, especially in bAP and mAP. Moreover, S-VAE achieves the best overall

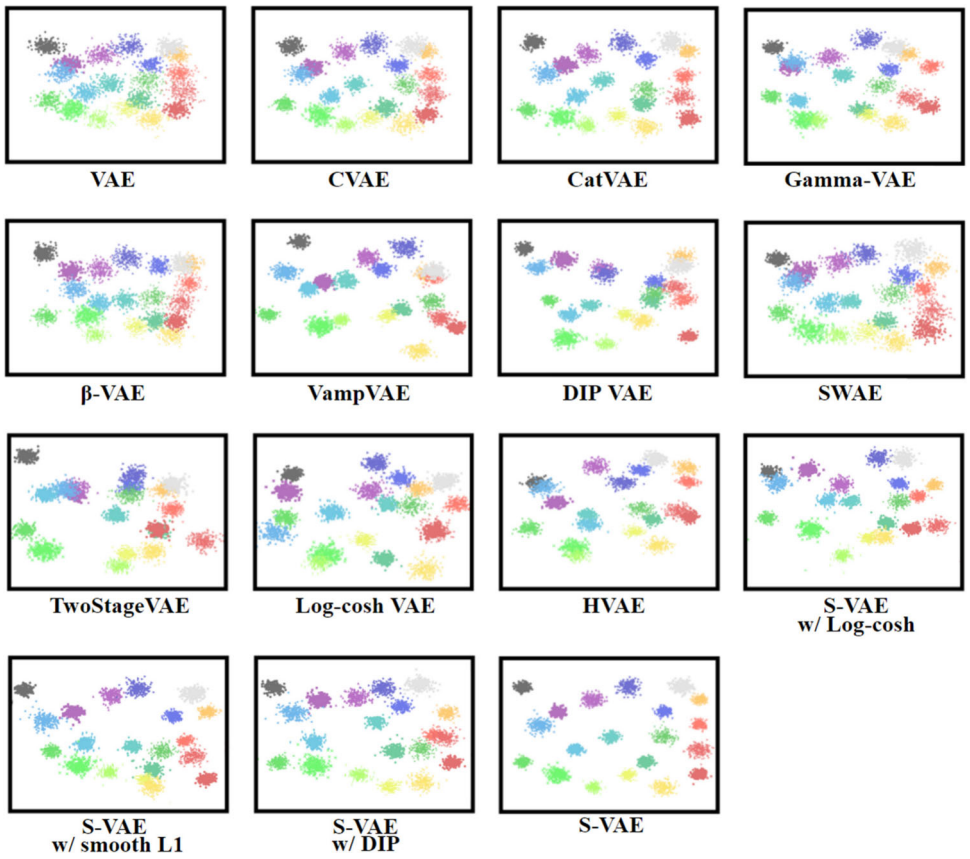



**Fig. 8** Feature space of features generated by VAE and its variants. The distribution of standard VAE and its variants in the feature space is visualized through t-SNE.

**Table 7** Performance comparison of detection heads

| Type | Method | mAP | bAP | nAP |
|---|---|---|---|---|
| No Improvement | TFA [8] | 62.64 | 67.05 | 49.40 |
| New Branch | DandR [44] | 65.59 | 70.37 | 51.25 |
| | CPE [45] | 64.47 | 68.95 | 51.03 |
| Optimization of | CT [46] | 63.79 | 70.09 | 44.90 |
| Structural | PCB [20] | 65.23 | 71.07 | 47.72 |
| Functionality | EDH | **68.07** | **73.14** | **52.85** |

results, indicating that the design of these three loss functions effectively improves model performance.

#### 4.5.3 Impact of Different Detection Heads on Detection Results

To investigate the impact of different detection heads on S-EDH-Faster RCNN's detection results, experiments were conducted using Disentangle and Remerge branches (DandR), Contrastive Proposal Encoding (CPE), Context-Transformer (CT), Prototypical Calibration Block (PCB), and the proposed Enhanced Detection Head. Table 7 shows the detection accuracies of S-EDH-Faster RCNN using different detection heads under the split1 with 10 shots scenario.

From Table 7, we can find that: The method of using the EDH achieved the best results, with mean average precisions for all classes, base classes and novel classes at 68.07%, 73.14%, and 52.85%, respectively. Compared to other methods, these figures represent improvements of 2.48%-5.43%, 2.07%-6.09%, and 1.60%-7.95%. The EDH outperforms others by utilizing CBAM to strengthen classification features, allowing the model to focus more on important regions, thereby improving classification performance. Additionally, by computing cosine similarity between enhanced RoI features and class prototype features and combining this with softmax scores, the detection head better integrates feature extraction and classifier judgment, ultimately enhancing the model's detection effectiveness.

#### 4.5.4 Impact of different fine-tuning strategies on detection results

To explore the effect of different fine-tuning strategies on the detection results of S-EDH-Faster RCNN, this subsection employs various fine-tuning strategies to train S-EDH-Faster RCNN. Table 8 and Fig. 9present the corresponding detection results.

From Table 8, it is evident that when using the adaptive fine-tuning method, S-EDH-Faster RCNN achieves an average accuracy of 68.07%, outperforming other strategies by 5.06%-6.62%. Additionally, the average precision for base classes and novel classes are 73.14% and 52.85%, respectively, which are 2.1%-6.15% and 3.45%-14.89% higher than those achieved using other strategies. This improvement can be attributed to the model dynamically learning the features of novel class samples by adaptively selecting important convolutional kernels in the backbone network, thereby demonstrating stronger generalization ability in few-shot scenarios.

### 4.6 Ablation studies on blurring, lighting, and occlusion condition

In order to critically evaluate the robustness of the model S-EDH-Faster RCNN, this subsection performs ablation studies under blurring, illumination, and occlusion conditions, with extensive ablation studies under a variety of challenging conditions. The purpose of these experiments is to understand the impact of each adverse condition on the model's detection performance, so as to gain insight into its practical applicability in real-world scenarios.

#### 4.6.1 Impact of different degrees of blurring on detection results

To explore the impact of blur on the detection results, we simulated different degrees of blurring using Gaussian blur with different kernel sizes and standard deviations $\sigma$. The magnitude relationship between Gaussian kernel and standard

**Table 8** Performance comparison of fine-tuning methods

| Type | Method | mAP | bAP | nAP |
|---|---|---|---|---|
| Fine-tune only | TFA [8] | 62.64 | 67.05 | 49.40 |
| the detection heads | Symmetric Fine-Tuning [48] | 61.45 | 66.99 | 44.83 |
| Fine-tune only | Bias-Balanced RPN [49] | 62.77 | 71.04 | 37.96 |
| fixed network layers | Online Inferential Calibration [50] | 63.01 | 67.77 | 48.73 |
| Dynamically fine-tune the backbone network | Adaptive Fine-Tuning | **68.07** | **73.14** | **52.85** |

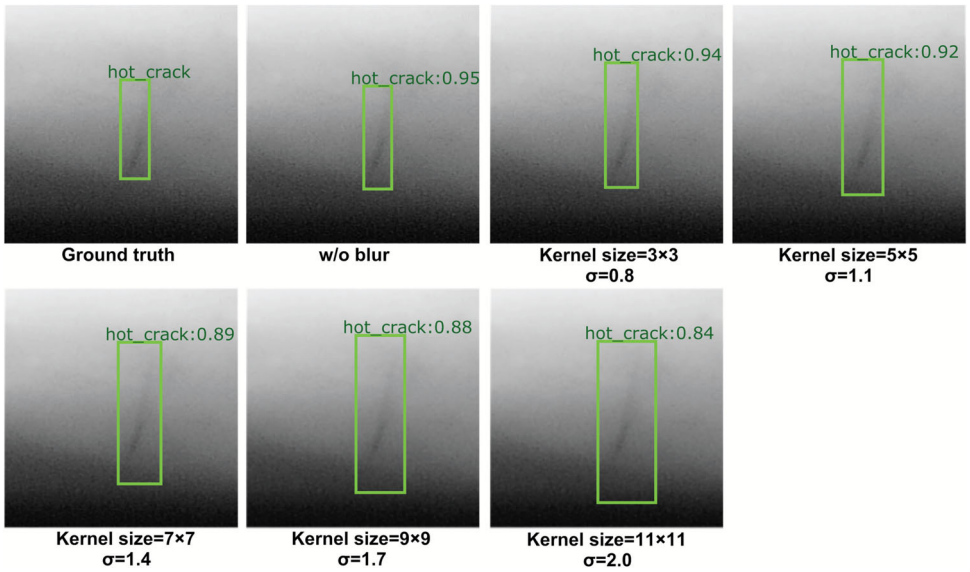



**Fig. 9** Visualization of the effect of Gaussian blur on object detection accuracy on PIP-DET split1 (10 shots)

deviation is: $\sigma = 0.3 \times ((\text{kernel} - 1) \times 0.5 - 1) + 0.8$. The 0.3 and 0.8 in the formula are empirical coefficients that are used to establish a reasonable mapping between the convolution kernel size and the standard deviation. Table 9 and Fig. 9 show the results of adding blurring to all training set images on PIP-DET dataset (taking split1-10 shots as an example).

From Table 9, it can be got that: Kernel=0, $\sigma$=0 (no blurring applied) yields the highest performance across all metrics, with mAP of 68.07%, bAP of 73.14%, and nAP of 52.85%. This serves as the baseline for comparing blurred combinations. Specifically, under slight blurring (kernel size=3×3, $\sigma$=0.8), the model maintained high performance, with mAP of 65.34%, bAP of 71.15%, and nAP of 47.90%. When the kernel size increased to 5 and the $\sigma$ was 1.1, the mAP decreased slightly to 62.16%, bAP to 67.26%, and nAP to 46.84%. The kernel size was further enhanced to 7 with $\sigma$=1.4, and the mAP decreased to 55.72%, bAP to 58.83%, and nAP to 46.37%. When the kernel size is 9×9 and the standard deviation is 1.7, the mAP decreases to 51.01%, bAP decreases to 52.11%, and nAP increases slightly to 47.70%. However, under the strongest blurring processing (kernel size=11×11, $\sigma$=2.0), the performance of the model decreased significantly, with mAP of 49.73%, bAP of 51.53%, and nAP of 44.30%. This demonstrates that with the increase of blurring, the detection accuracy of the model generally decreased, especially in the case of high blurring.

**Table 9** Impact of Gaussian Blur on object detection accuracy on PIP-DET split1 (10 shots)

| Kernel Size | $\sigma$ | mAP | bAP | nAP |
|---|---|---|---|---|
| 0×0 | 0 | 68.07 | 73.14 | 52.85 |
| 3×3 | 0.8 | 65.34 | 71.15 | 47.90 |
| 5×5 | 1.1 | 62.16 | 67.26 | 46.84 |
| 7×7 | 1.4 | 55.72 | 58.83 | 46.37 |
| 9×9 | 1.7 | 51.01 | 52.11 | 47.70 |
| 11×11 | 2.0 | 49.73 | 51.53 | 44.30 |

### 4.6.2 Impact of different degrees of lighting on detection results

To analyze the impact of light on the detection results, we construct experiments with different brightness levels. Table 10 and Fig. 10 present the detection performance across five different brightness levels on PIP-DET dataset (using split1-10 shots as an example).

From Table 10 and Fig. 10, it can be find that: the highest detection performance is achieved under normal lighting conditions (Brightness=0), with an mAP of 68.07%, bAP of 73.14%, and nAP of 52.85%. This serves as the baseline for comparison with other brightness settings.

Under moderate brightness reduction (Brightness=-25), the model maintains relatively high performance, with an mAP of 66.28%, bAP of 72.00%, and nAP of 49.12%. However, with a more extreme reduction in brightness (Brightness=-50), detection accuracy drops significantly,

**Table 10** Impact of brightness changes on object detection accuracy on PIP-DET split1 (10 shots)

| Brightness | mAP | bAP | nAP |
|---|---|---|---|
| -50 | 64.38 | 70.80 | 45.12 |
| -25 | 66.28 | 72.00 | 49.12 |
| 0 | 68.07 | 73.14 | 52.85 |
| 25 | 66.25 | 71.58 | 50.25 |
| 50 | 65.85 | 72.44 | 46.09 |

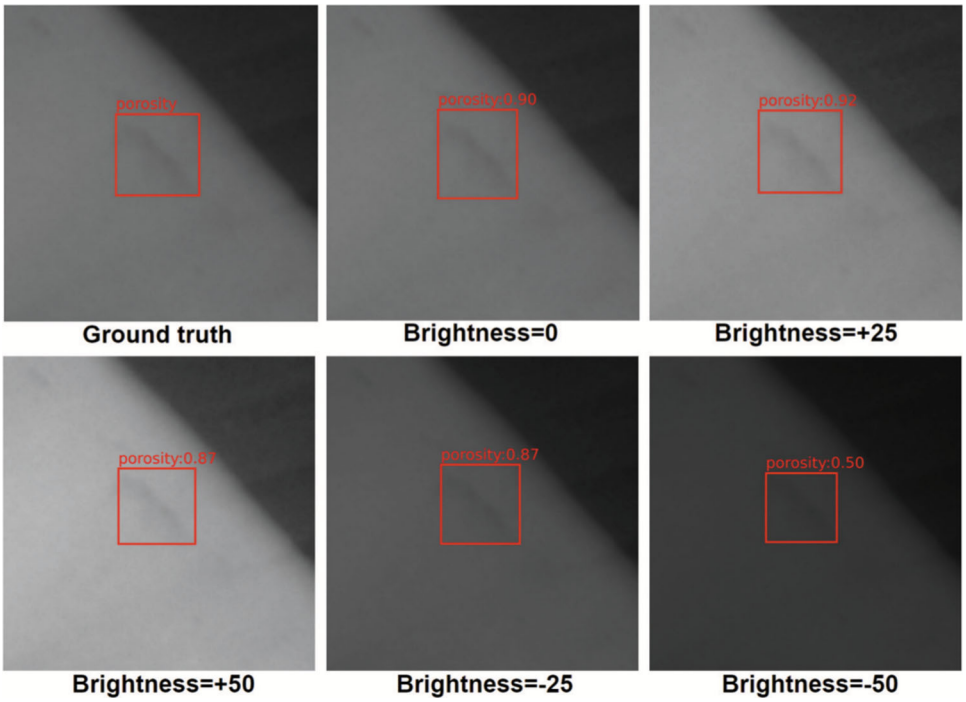



**Fig. 10** Visualization of the effect of brightness on object detection accuracy on PIP-DET split1 (10 shots)

with an mAP of 64.38%, bAP of 70.80%, and nAP of 45.12%. This suggests that under darker conditions, the model struggles to extract useful features, leading to performance degradation.

Similarly, increasing brightness to 25 and 50 leads to a slight decline in performance. At Brightness=25, it got 66.25% mAP, 71.58% bAP, and 50.25% nAP. With an even higher brightness level (Brightness=50), the mAP decreases to 65.85%, bAP to 72.44%, and nAP to 46.09%. This indicates that excessive brightness can also impact detection performance, likely due to overexposure causing loss of important details.

Overall, this demonstrate that while moderate lighting changes have minimal impact, extreme lighting conditions may reduce detection accuracy, with darker conditions having a more significant negative effect.

#### 4.6.3 Impact of Different Degrees of Occlusion on Detection Results

To evaluate the robustness of the model against occlusion, we introduced occlusion patches covering 5%, 10%, 15%, 20%, and 25% of the image area during training. These occlusions are simulated with randomly placed rectangular patches of varying aspect ratios, mimicking real-world dust, lens smudges, and occlusion projections. Table 11 and Fig. 11 present the detection performance across different occlusion levels on PIP-DET dataset (using split1-10 shots as an example).

Tables 11 and Fig. 11 present the detection performance across different occlusion levels on PIP-DET dataset (using split1-10 shots as an example).

The results indicate that detection performance decreases as occlusion levels increase. The highest accuracy is observed under no occlusion (Occlusion=0%), with an mAP of 68.07%, bAP of 73.14%, and nAP of 52.85%, serving as the baseline for comparison.

Under minor occlusion (5%), the model maintains relatively high performance, with an mAP of 65.89%, bAP of 72.57%, and nAP of 45.85%. However, as the occlusion level increases to 10%, mAP decreases to 65.24%, and nAP drops to 43.46%, showing a clear negative impact on detecting novel objects.

With moderate occlusion (15-20%), the performance further declines. At 15% occlusion, the mAP drops to 64.79%, and nAP decreases to 41.64%. At 20%, mAP declines slightly

**Table 11** Impact of occlusion on object detection accuracy on PIP-DET split1 (10 shots)

| Occlusion Percentage | mAP | bAP | nAP |
|---|---|---|---|
| 5% | 65.89 | 72.57 | 45.85 |
| 10% | 65.24 | 72.50 | 43.46 |
| 15% | 64.79 | 72.51 | 41.64 |
| 20% | 64.51 | 72.57 | 40.33 |
| 25% | 64.15 | 72.50 | 39.10 |
| 0% (No Occlusion) | 68.07 | 73.14 | 52.85 |

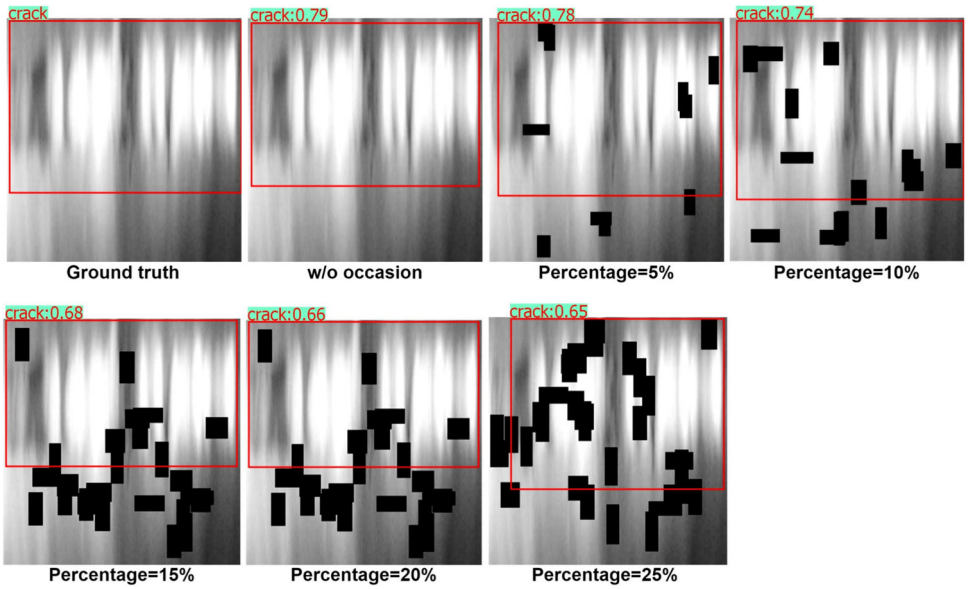



**Fig. 11** Visualization of the effect of brightness on object detection accuracy on PIP-DET split1 (10 shots)

to 64.51%, and nAP drops to 40.33%, highlighting the difficulty in identifying partially obscured objects.

When occlusion reaches 25%, the detection accuracy declines significantly, with mAP falling to 64.15%, bAP to 72.50%, and nAP to 39.10%, demonstrating that high occlusion levels severely hinder the model's ability to recognize novel objects.

The overall trend suggests that as occlusion increases, detection accuracy, especially for novel objects (nAP). While the model shows some robustness to minor occlusions (less than 10%), larger occlusions (more than 15%) significantly degrade performance, likely due to missing key visual features required for recognition.

### 4.7 Comparison on the public NEU-DET dataset

To validate the performance of the proposed method on a public dataset, this subsection selects the NEU-DET dataset to compare the detection performance of different methods. The experiments were conducted under different numbers of training samples (1, 2, 3, 5 and 10 shots).

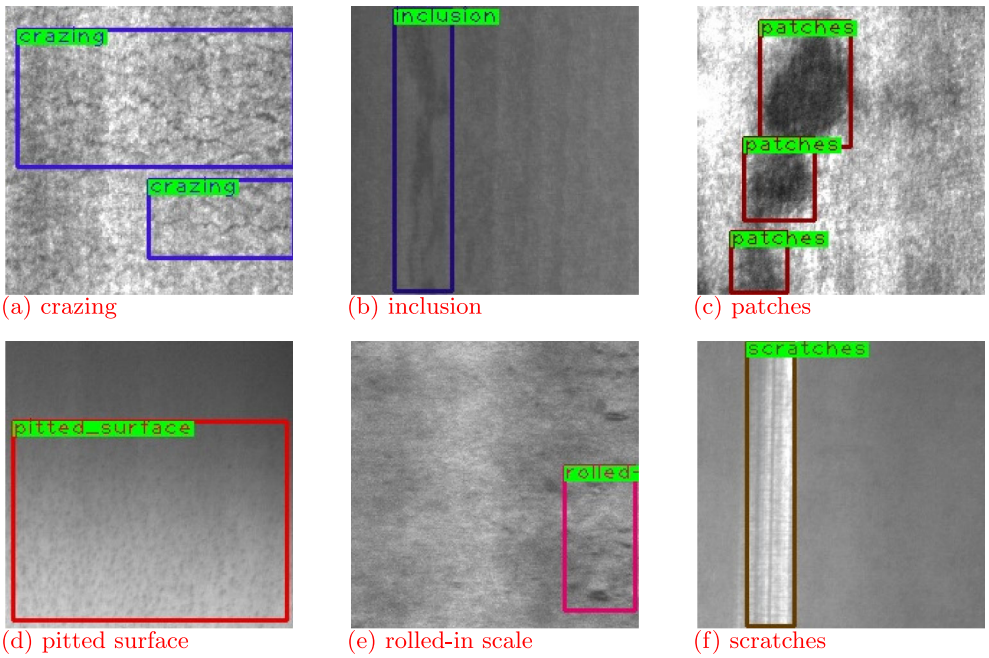
NEU-DET is a dataset for steel surface defect detection, containing six types of defects with 300 samples each, totaling 1,800 samples and corresponding labels [56]. The dataset is divided into 1,260 training samples and 540 test samples. Figure 12 shows annotation examples of defect images. In this experiment, four classes were randomly selected as base classes, and the remaining two classes were defined as novel classes. Each novel class had K=1, 2, 3, 5, 10 annotated training samples to simulate few-shot scenarios. This study considered three random splits, named split1, split2 and split3. During training and testing, all images in the dataset were resized to 200×200 with 3 channels.

Table 12 shows the average accuracy of different methods on novel classes, where bold numbers indicate the best results and underlined numbers indicate the second-best results. Figure 13 showcases detection examples of different methods. From Table 12 and Fig. 11, it can be concluded that: The proposed method outperforms other methods overall across all splits and shots. Specifically, in split1, the proposed method achieved 15.72% nAP with only one sample. As the number of samples increased, the nAP significantly improved, reaching 38.21% with ten samples. In split2, the proposed method achieved the highest nAP for all sample sizes. Similarly, in split3, the proposed method achieved the highest average accuracy in most cases. Additionally, from the detection example images, the proposed methods detection results closely matched the true annotations. Other methods, such as the CT method, exhibited false positives, while the VFA method showed multiple overlapping bounding boxes. This indicates that the proposed method can accurately detect and locate objects in few-shot scenarios.

### 4.8 Comparison on the public PCB dataset

In order to validate the performance of the proposed method on other material datasets and in different scenes, this subsection selects the PCB dataset [57] to compare the detection performance of different methods. The experiments were conducted under different numbers of training samples (1, 2, 3, 5 and 10 shots).

This dataset is specifically designed for defect detection in printed circuit boards and consists of a total of 12,428 images across six defect categories: missing hole, mouse bite, open circuit, short, spur and spurious copper. The detail detection results of different methods are provided in Table 13.




(a) crazing (b) inclusion (c) patches

(d) pitted surface (e) rolled-in scale (f) scratches

**Fig. 12** Examples of defect images with annotations on NEU-DET

Figure 14 shows annotation examples of defect images. In this experiment, four classes were randomly selected as base classes, and the remaining two classes were defined as novel classes. Each novel class had K=1, 2, 3, 5, 10 annotated training samples to simulate few-shot scenarios. We considered three random splits, named split1, split2 and split3.

Table 13 shows the average accuracy of different methods on novel classes in the PCB dataset, where bold numbers indicate the best results and underlined numbers indicate the second-best results. From Table 13, it can be concluded that: The proposed method outperforms other methods overall across all splits and shots. Specifically, in split1, the proposed method achieved 34.75% nAP with only one sample. As the number of samples increased, the nAP significantly improved, reaching 56.65% with ten samples. In split2, the proposed method achieved the highest nAP for all sample sizes, with a notable improvement from 32.12% with one sample to 52.44% with ten samples. Similarly, in split3, the proposed method achieved the highest average accuracy in most cases, starting at 30.68% with one sample and reaching 54.74% with ten samples. Figure 15 shows the detection results of the different methods.

The proposed method's consistent superiority across all splits and sample sizes demonstrates its robustness and

**Table 12** Experimental results (nAP) of different methods on NEU-DET dataset

| Method | Novel Split1 | | | | | Novel Split2 | | | | | Novel Split3 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 5 | 10 | 1 | 2 | 3 | 5 | 10 | 1 | 2 | 3 | 5 | 10 |
| CME [23] | 8.47 | 9.27 | 11.69 | 16.22 | 22.69 | 4.49 | 5.63 | 8.47 | 10.82 | 16.64 | 3.68 | 3.99 | 8.75 | 8.87 | 17.19 |
| FSCE [45] | 12.25 | 15.72 | 18.68 | 25.74 | 33.15 | 8.57 | 10.47 | 12.47 | 20.42 | 29.19 | 4.74 | 10.21 | 11.69 | 19.74 | 26.86 |
| TFA [8] | 9.93 | 12.03 | 18.05 | 20.56 | 30.68 | 7.51 | 9.19 | 12.99 | 16.49 | 26.63 | 5.83 | 7.01 | 10.95 | 19.64 | 25.12 |
| Meta Faster RCNN [18] | 6.45 | 7.77 | 10.41 | 15.21 | 20.93 | 2.75 | 4.14 | 7.46 | 10.02 | 17.29 | 2.67 | 3.09 | 4.08 | 8.34 | 11.91 |
| DandR [44] | 15.42 | 16.21 | 19.25 | 26.89 | 35.26 | 9.35 | 11.22 | 14.78 | 20.49 | 30.52 | 7.07 | 10.56 | 12.38 | 18.40 | 27.80 |
| VFA [19] | 11.33 | 16.41 | 17.91 | 23.94 | 32.54 | 8.63 | 10.79 | 15.32 | 20.95 | 31.22 | 7.23 | 9.11 | 8.97 | 18.62 | 27.77 |
| DeFRCN [20] | 10.68 | 11.59 | 19.01 | 24.01 | 31.52 | 7.26 | 10.29 | 15.69 | 18.37 | 28.52 | 5.86 | 7.52 | 9.92 | 17.44 | 24.43 |
| CT [46] | 7.97 | 9.79 | 11.53 | 17.23 | 21.63 | 3.15 | 5.25 | 9.65 | 9.73 | 17.52 | 3.66 | 4.42 | 5.41 | 9.09 | 15.31 |
| RegAD [59] | 5.91 | 8.52 | 12.29 | 20.52 | 27.85 | 2.72 | 5.65 | 8.43 | 16.71 | 24.21 | 1.26 | 2.68 | 7.22 | 15.46 | 22.59 |
| Ours | **15.72** | **17.39** | **20.47** | **29.51** | **38.21** | **9.42** | **12.72** | **19.16** | **22.56** | **35.13** | **8.18** | **11.99** | **15.75** | **23.34** | **30.32** |

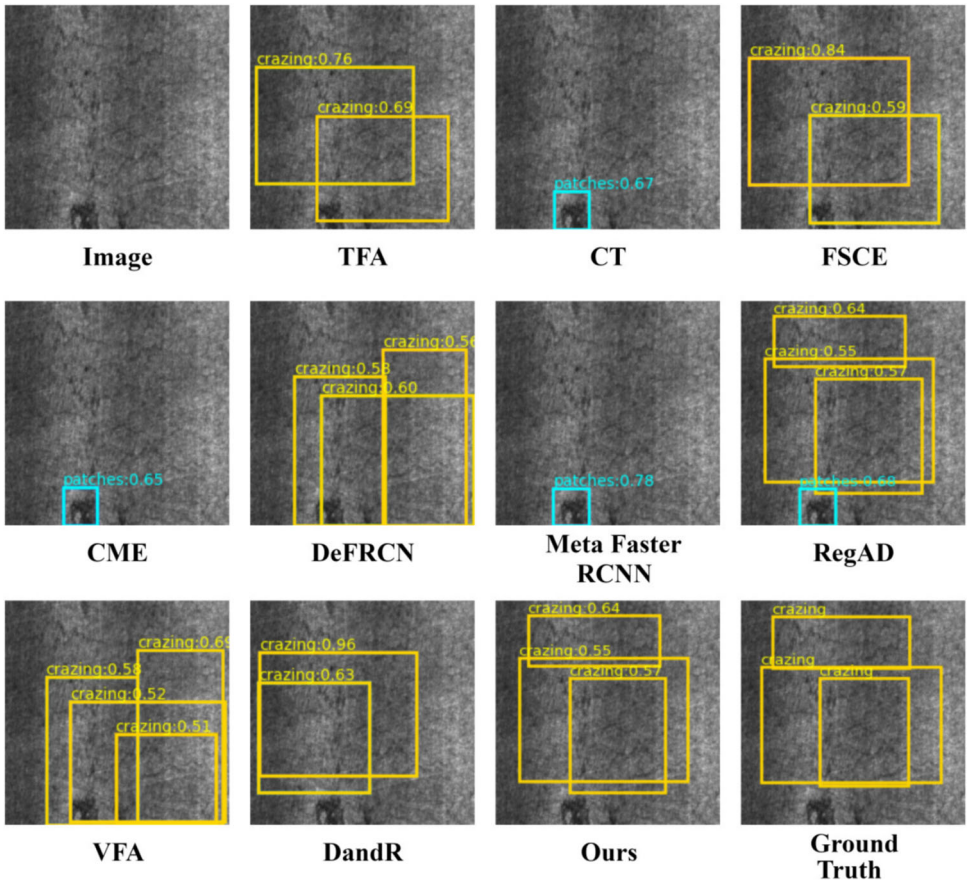



**Fig. 13** Detection examples of different methods on NEU-DET split1 (10 shots)

effectiveness in few-shot scenarios. Other methods, such as DeFRCN and DandR, also showed competitive performance but were consistently outperformed by the proposed method. For instance, in split1, DeFRCN achieved 53.78% nAP with ten samples, which is close but still lower than the proposed method's 56.65%. Similarly, in split2, DandR achieved 47.33% nAP with ten samples, which is significantly lower than the proposed method's 52.44%. These results highlight the proposed method's ability to generalize well across different splits and sample sizes, making it a strong candidate for few-shot object detection tasks on PCB dataset.

**Table 13** Experimental results (nAP) of different methods on PCB dataset

| Method | Novel Split1 | | | | | Novel Split2 | | | | | Novel Split3 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 5 | 10 | 1 | 2 | 3 | 5 | 10 | 1 | 2 | 3 | 5 | 10 |
| CME [23] | 27.04 | 31.64 | 36.03 | 39.72 | 42.35 | 22.42 | 24.37 | 34.17 | 35.02 | 38.74 | 19.66 | 24.44 | 32.37 | 33.13 | 41.95 |
| FSCE [45] | 29.48 | 35.63 | 39.51 | 45.07 | 51.87 | 19.57 | 22.14 | 27.91 | 34.5 | 43.25 | 24.01 | 29.71 | 34.14 | 42.22 | 50.48 |
| TFA [8] | 27.27 | 29.36 | 37.66 | 44.09 | 50.52 | 23.24 | 26.92 | 31.78 | 30.51 | 34.00 | 23.12 | 27.82 | 34.39 | 39.96 | 40.46 |
| Meta Faster RCNN [18] | 9.30 | 17.70 | 19.51 | 20.69 | 25.99 | 7.45 | 10.74 | 11.24 | 14.22 | 18.30 | 9.60 | 14.39 | 16.44 | 17.87 | 24.19 |
| DandR [44] | 32.46 | 35.86 | 39.93 | 44.17 | 51.78 | 29.49 | 31.32 | 36.01 | 44.89 | 47.33 | 24.82 | 28.93 | 35.86 | 40.41 | 42.76 |
| VFA [19] | 27.05 | 32.70 | 35.23 | 39.28 | 44.59 | 23.65 | 30.70 | 35.94 | 39.82 | 44.24 | 28.53 | 31.83 | 38.03 | 42.88 | 47.38 |
| DeFRCN [20] | 31.22 | 38.75 | **44.64** | 47.55 | 53.78 | 28.83 | 32.07 | 38.74 | 42.70 | 50.44 | 28.82 | 34.38 | 39.79 | 44.05 | 50.82 |
| CT [46] | 7.74 | 10.00 | 16.23 | 17.23 | 25.69 | 7.92 | 12.45 | 17.49 | 20.24 | 33.73 | 7.16 | 12.23 | 16.91 | 19.35 | 28.57 |
| RegAD [59] | 24.42 | 31.44 | 35.22 | 40.37 | 46.78 | 19.99 | 24.19 | 27.33 | 33.16 | 40.06 | 21.81 | 23.90 | 28.60 | 32.96 | 42.29 |
| Ours | **34.75** | **39.75** | 44.56 | **49.04** | **56.65** | **32.12** | **35.11** | **41.15** | **45.12** | **52.44** | **30.68** | **39.62** | **40.18** | **46.35** | **54.74** |

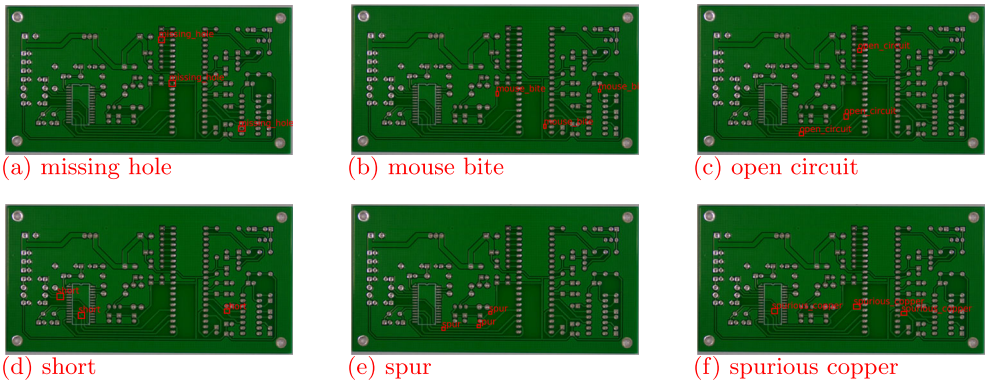



**Fig. 14** Examples of defect images with annotations on PCB datasets

### 4.9 Comparison on the public MT defect dataset

In order to validate the performance of the proposed method on other material datasets and different scenes, this subsection selects the MT defect dataset [58] to compare the detection performance of different methods. The experiments were conducted under different numbers of training samples (1, 2, 3, 5 and 10 shots).

In the MT defect dataset, the images of 5 common magnetic tile defects and a no-defect class, totaling 393 images, were collected, and their pixel level ground-truth were labeled. Figure 16 shows annotation examples of defect images. In order to adapt to the detection task, we converted the mask annotations of the segmentation task into the annotations of object detection. In this experiment, four classes were randomly selected as base classes, and the remaining two classes were defined as novel classes. Each novel class had K=1, 2, 3, 5, 10 annotated training samples to simulate few-shot scenarios. This study considered three random splits, named split1, split2 and split3.

From the comparison results in Table 14 and Fig 17, it can be seen that the proposed method achieves the best

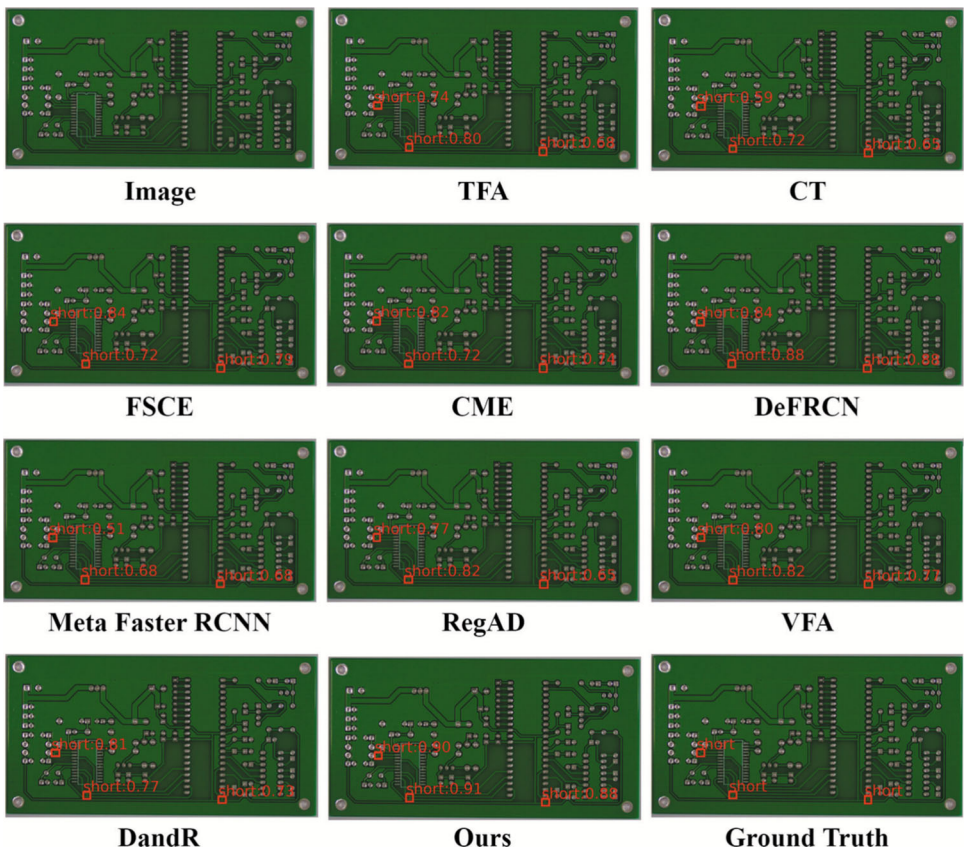
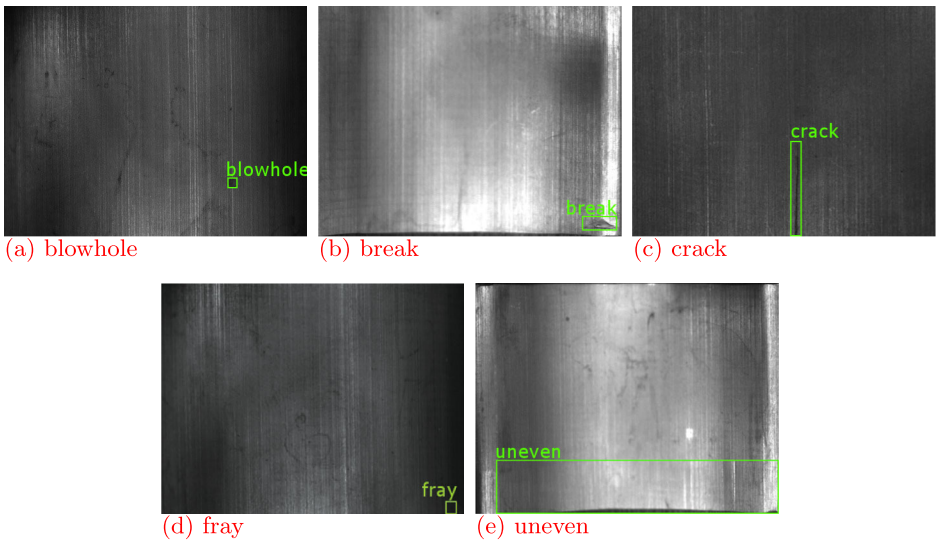



**Fig. 15** Detection examples of different methods on PCB split1 (10 shots)




(a) blowhole (b) break (c) crack (d) fray (e) uneven

**Fig. 16** Examples of defect images with annotations on MT split1 (10 shots)

performance across all splits and different sample sizes, significantly outperforming other methods. For example, in split1, the method reaches an nAP of 16.96% with only one sample, and its performance further improves as the number of samples increases, reaching 40.72% in the 10-shot setting. In split2 and split3, the method also maintains a leading position in most cases. In the 10-shot setting of split1, split2, and split3, the proposed method outperforms the second-best method by 3.59%, 4.03%, and 2.62%, respectively, further demonstrating its robustness and superiority. From the detection example images, the results of the proposed method align well with the ground truth annotations, indicating its stronger object detection capability in few-shot scenarios.

### 4.10 The computational complexity of different methods

The Table 15 lists the impact of different approaches on model size (#param) and FLOPs. From these results, it can be seen that the number of parameters of our method is 59.3M, and the number of floating-point operations is 18.7G.

## 5 Discussion

Although the model demonstrates a high level of performance in detecting novel classes of defects, there remain instances of extremely challenging or complex defect morphologies that result in over-detection or under-detection issues. Figure 18 shows examples of over-detection and under-detection on PIP-DET dataset. In cases of over-detection, the model excessively interprets irrelevant features in the background, mistakenly identifying additional areas as defective. In scenarios involving multiple defects, the model fails to recognize all the genuine defects present. This indicates that when detecting novel class samples, if the samples lack enough representativeness or the defects are inherently difficult to detect, the model may be prone to over-detection or under-detection.

Figure 19 (a) reveals the mean value of the original feature distribution at 0.3389, with a standard deviation of 0.4547, and a peak value reaching 6.1654. Figure 19 (b) shows that the feature distribution generated by S-VAE closely aligns with the original, registering a mean value of 0.3313, a standard deviation of 0.4445, and a maximum of 4.9915, closely approximating the initial data while maintaining a similar histogram shape. Figure 19 (c) displays the reconstructed feature maps post-application of a blur with kernel size $21 \times 21$ and a standard deviation of 3.5, noting a reduction in the mean feature distribution to 0.3145, a decrease in standard deviation to 0.4283, and a reduced peak of 3.4113. Similarly, Fig. 19 (d) presents the reconstructed feature maps following the introduction of 20% salt-and-pepper noise to the original image, marking a substantial rise in the mean to 0.7745, an escalation in standard deviation to 0.9235, and a peak value of 6.9007. Figure 20 illustrates the characterization of the first eight channels of layer 2.

This visualization in Fig. 21 illustrates the failures of Medoids in a noisy environment. Specifically, when there are too many outlier points, it will bring high noise to the calculation, causing the Medoids to choose the features that deviate from the main cluster as the feature prototype. Since Medoids must select data points as the center, in a high-noise environment, it may mistakenly select some features that are far away from the real cluster, or even choose noise features as the center, resulting in unreasonable calculation results for feature prototypes. Although using Medoids can reduce the impact of outliers, it may still not completely eliminate the negative effects of these factors, especially when the feature distribution is more dispersed or there is a lot of noise.

**Table 14** Experimental results (nAP) of different methods on MT defect dataset

| Method | Novel Split1 | | | | | Novel Split2 | | | | | Novel Split3 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 5 | 10 | 1 | 2 | 3 | 5 | 10 | 1 | 2 | 3 | 5 | 10 |
| CME [23] | 8.93 | 9.92 | 12.43 | 17.39 | 23.93 | 4.75 | 6.03 | 8.97 | 11.49 | 17.49 | 3.93 | 4.28 | 9.41 | 9.56 | 18.07 |
| FSCE [45] | 12.98 | 16.84 | 19.68 | 27.74 | 35.65 | 9.18 | 11.00 | 13.23 | 21.59 | 31.10 | 5.01 | 10.95 | 12.33 | 20.92 | 28.99 |
| TFA [8] | 10.69 | 12.69 | 19.11 | 21.61 | 32.27 | 8.11 | 9.88 | 13.81 | 17.35 | 28.48 | 6.21 | 7.56 | 11.78 | 21.06 | 26.73 |
| Meta Faster RCNN [18] | 6.80 | 8.37 | 11.13 | 16.15 | 22.58 | 2.91 | 4.43 | 7.87 | 10.69 | 18.16 | 2.81 | 3.27 | 4.36 | 8.98 | 12.65 |
| DandR [44] | 16.21 | 17.43 | 20.35 | 28.57 | 37.13 | 10.02 | 11.92 | 15.87 | 21.93 | 32.28 | 7.53 | 11.36 | 13.17 | 19.63 | 29.40 |
| VFA [19] | 12.18 | 17.61 | 19.33 | 25.18 | 34.33 | 9.25 | 11.37 | 16.13 | 22.53 | 33.03 | 7.76 | 9.83 | 9.59 | 19.95 | 29.52 |
| DeFRCN [20] | 11.32 | 12.18 | 20.12 | 25.72 | 33.69 | 7.82 | 11.00 | 16.89 | 19.61 | 30.45 | 6.28 | 7.90 | 10.60 | 18.56 | 26.00 |
| CT [46] | 8.44 | 10.50 | 12.23 | 18.12 | 23.16 | 3.39 | 5.60 | 10.32 | 10.24 | 18.66 | 3.95 | 4.73 | 5.71 | 9.55 | 16.25 |
| RegAD [59] | 6.28 | 9.15 | 13.05 | 21.78 | 30.03 | 2.89 | 6.08 | 9.05 | 17.97 | 25.56 | 1.33 | 2.84 | 7.59 | 16.60 | 24.22 |
| Ours | **16.96** | **18.64** | **22.08** | **31.12** | **40.72** | **10.09** | **13.55** | **20.14** | **23.83** | **37.06** | **8.79** | **12.67** | **16.77** | **24.95** | **32.14** |

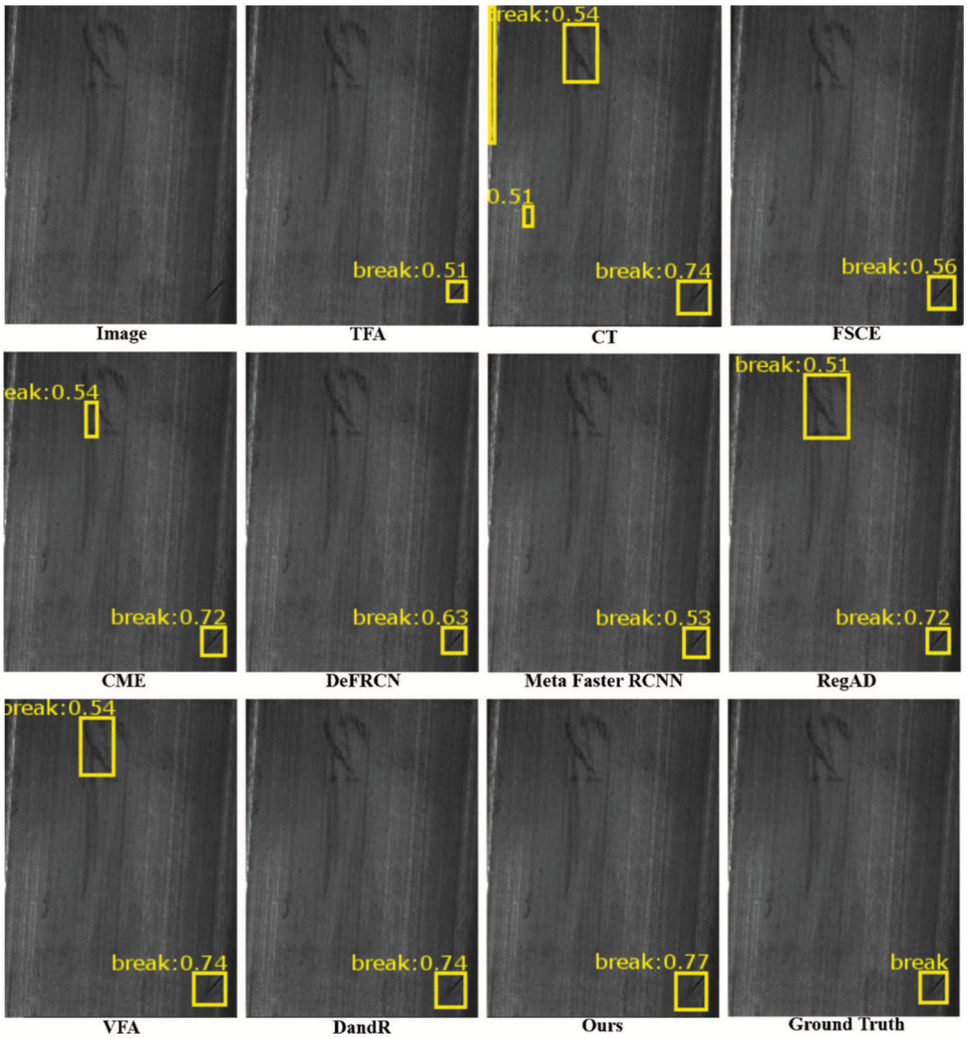



**Fig. 17** Detection examples of different methods on MT split1 (10 shots)

## 6 Conclusion

This paper proposes a Faster RCNN-based method for small-sample pipeline DR defect detection, incorporating a smooth variational autoencoder and enhanced detection head. The main contributions of this work include the following three aspects: First, to alleviate the issue of scarce novel class training samples, a smooth variational autoencoder is used to reconstruct features during DR defect image feature extraction, better fitting the distribution of training data. Second, to enhance the accuracy of classification tasks, an enhanced detection head is designed, employing a CBAM-based center point classification calibration module to correct classification scores, thereby improving classification precision. Finally, to better learn the characteristics of novel class samples, an adaptive fine-tuning method is proposed, adaptively updating key convolutional kernels in the backbone network during the fine-tuning stage, enabling the model to generalize better to novel classes. Experimental results on both the self-made dataset and public datasets demonstrate that the proposed method outperforms others in terms of average accuracy, proving its superiority.

However, our method still has limitations. As pointed in Discussion, its performance is affected by the generated

**Table 15** Computational complexity of different methods

| Method | #param. (M) | FLOPs (G) |
|---|---|---|
| CME [23] | 60.1 | 15.3 |
| FSCE [45] | 58.6 | 15.7 |
| TFA [8] | 55.5 | 13.4 |
| Meta Faster RCNN [18] | 62.6 | 16.5 |
| DandR [44] | 66.5 | 19.4 |
| VFA [19] | 60.3 | 15.4 |
| DeFRCN [20] | 58.3 | 16.1 |
| CT [46] | 60.0 | 15.7 |
| RegAD [59] | 78.5 | 22.3 |
| Ours | 59.3 | 18.7 |

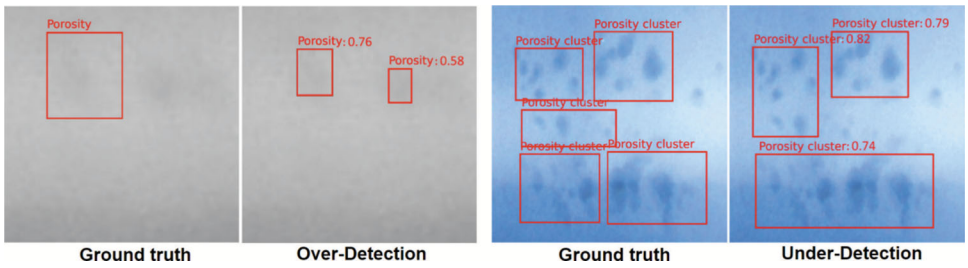
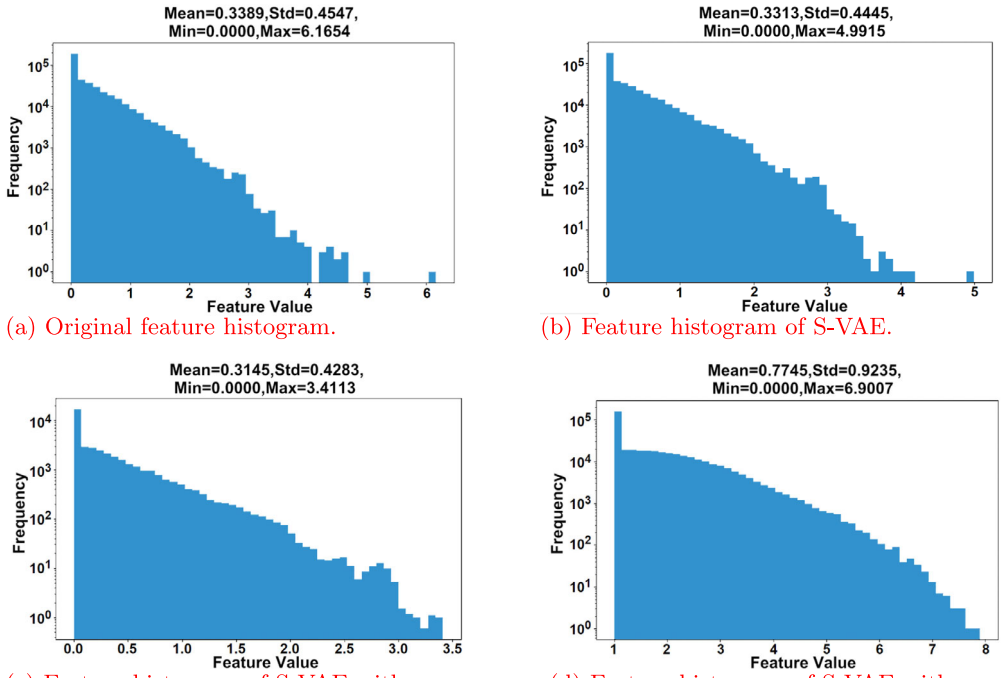



**Fig. 18** Examples of over-detection and under-detection on PIP-DET dataset



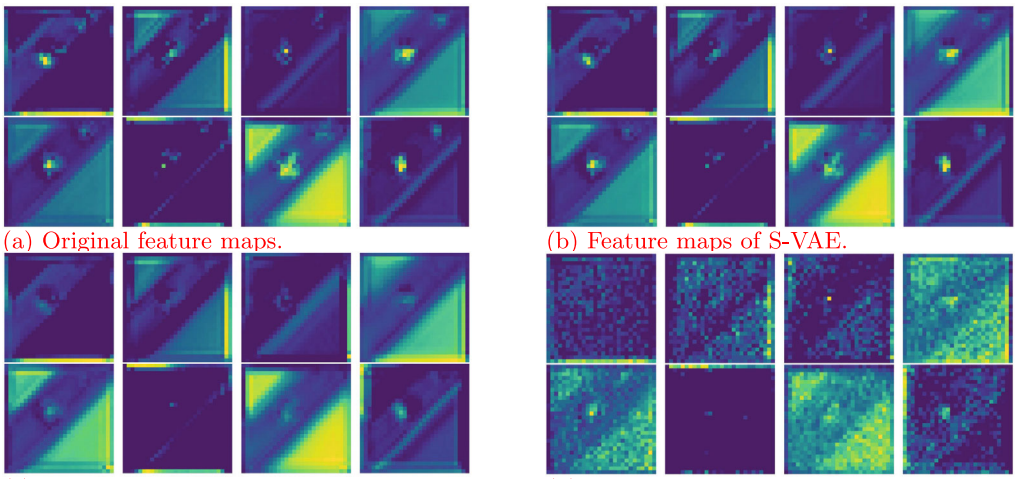

(a) Original feature histogram.

(b) Feature histogram of S-VAE.

(c) Feature histogram of S-VAE with Gaussian blur.

(d) Feature histogram of S-VAE with salt-and-pepper noise.

**Fig. 19** Histogram of S-VAE features under high-intensity salt-and-pepper noise and Gaussian blur



(a) Original feature maps.

(b) Feature maps of S-VAE.

(c) Feature maps of S-VAE with Gaussian blur.

(d) Feature maps of S-VAE with salt-and-pepper noise.

**Fig. 20** Examples of S-VAE failures under high-intensity salt-and-pepper noise and Gaussian blur

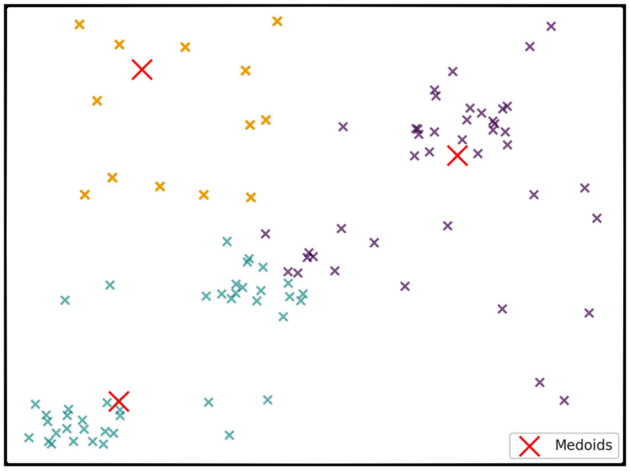



**Fig. 21** Examples of Medoids method failures in high-noise situations. In 2D simulation experiments, when the features are highly discrete, it is difficult for the Medoids method to select the truly representative feature prototypes

features. Therefor, in the future, one direction is to explore advanced generation methods, to better adapt to different types of defects or datasets. Furthermore, integrating self-supervised or semi-supervised learning techniques could enhance its ability to generalize to datasets with limited labeled samples, such as rare defect detection or high-resolution satellite image analysis.

**Acknowledgements** This work is supported by the National Natural Science Foundation of China (61906005, 61806013, 61876010, 62166002, 62176009), Natural Science Foundation of Xinjiang Uygur Autonomous Region (2023D01A22), General Project of Science and Technology Plan of Beijing Municipal Education Commission (KM202110005028), International Research Cooperation Seed Fund of Beijing University of Technology (2021A01), and Foundation Project of the Science.



## References

1. Yu Z, Wu Y, Wei B, Ding Z, Luo F (2023) A lightweight and efficient model for surface tiny defect detection. Appl Intell 53(6):6344–6353
2. Zheng Y, Cui L (2023) Defect detection on new samples with siamese defect-aware attention network. Appl Intell 53(4):4563–4578
3. Huang L, Gong A (2024) Surface defect detection for no-service rails with skeleton-aware accurate and fast network. IEEE Trans Indust Inf 23(3):4571–4581
4. Liu T, He Z, Lin Z, Cao G, Su W, Xie S (2024) An adaptive image segmentation network for surface defect detection. IEEE Trans Neural Networks Learn Syst 35(6):8510–8523
5. Gong Y, Liu M, Wang X, Liu C, Hu J (2024) Few-shot defect detection using feature enhancement and image generation for manufacturing quality inspection. Appl Intell 54(1):375–397
6. Meng Y, Xu H, Ma Z, Zhou J, Hui D (2023) Detail-semantic guide network based on spatial attention for surface defect detection with fewer samples. Appl Intell 53(6):7022–7040
7. Qi D, Hu J, Shen J (2023) Few-shot object detection with self-supervising and cooperative classifier. IEEE Trans Neural Networks Learn Syst 35(4):5435–5446
8. Ren X, Lin W, Yang X, Yu X, Gao H (2023) Data augmentation in defect detection of sanitary ceramics in small and non-i.i.d datasets. IEEE Trans Neural Networks Learn Syst 34(11): 8669-8678
9. Xu J, He J, Liu B, Cao F, Xiao Y (2024) A two-generation based method for few-shot learning with few-shot instance-level privileged information. Appl Intell 54(5):4077–4094
10. Song B, Zhu H, Bi Y (2024) A conditioned feature reconstruction network for few-shot classification. Appl Intell 54(8):6592–6605
11. Wang X, Huang TE, Darrell T, Darrell T, Gonzalez JE, Yu F (2020) Frustratingly simple few-shot object detection. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 1-15
12. Vettoruzzo A, Bouguelia MR, Vanschoren J, Rögnvaldsson T, Santosh KC (2024) Advances and challenges in meta-learning: A technical review. IEEE Trans Pattern Anal Mach Intell 46(7):4763–4779
13. Wang YX, Ramanan D, Hebert M (2019) Meta-learning to detect rare objects. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 9925-9934
14. Lu X, Diao W, Mao Y, Li J, Wang P, Sun X, Fu K (2023) Breaking immutable: Information-coupled prototype elaboration for few-shot object detection. In: Proceedings of the AAAI conference on artificial intelligence, pp 1844-1852
15. Wu X, Sahoo D, Hoi S (2020) Meta-rcnn: Meta learning for few-shot object detection. In: Proceedings of the ACM international conference on multimedia, pp 1679-1687
16. Cheng M, Wang H, Long Y (2021) Meta-learning-based incremental few-shot object detection. IEEE Trans Circuits Syst Video Technol 32(4):2158–2169
17. Demirel B, Baran O.B, Cinbis R.G (2023) Meta-tuning loss functions and data augmentation for few-shot object detection. In: Proceedings of the international conference on computer vision and pattern recognition, pp 7339-7349
18. Han G, Huang S, Ma J, He Y, Chang SF (2022) Meta Faster R-CNN: Towards accurate few-shot object detection with attentive feature alignment. In: Proceedings of the AAAI conference on artificial intelligence, pp 780-789
19. Han J, Ren Y, Ding J, Yan K, Xia GS (2023) Few-shot object detection via variational feature aggregation. In: Proceedings of the AAAI conference on artificial intelligence, pp 755-763
20. Qiao L, Zhao Y, Li Z, Qiu X, Wu J, Zhang C (2021) Defrcn: Decoupled Faster R-CNN for few-shot object detection. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 8681-8690
21. Choi T M, Kim J H (2023) Incremental few-shot object detection via simple fine-tuning approach. In: Proceedings of the IEEE international conference on robotics and automation, pp 9289-9295

22. Xin Z, Chen S, Wu T, Shao Y, Ding W, You X (2024) Few-shot object detection: Research advances and challenges. Inf Fus 102307
23. Li B, Yang B, Liu C, Liu F, Ji R, Ye Q (2021) Beyond max-margin: Class margin equilibrium for few-shot object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7363-7372
24. Xu J, Le H, Samaras D (2023) Generating Features with Increased Crop-related Diversity for Few-Shot Object Detection. In: Proceedings of the IEEE/CVF international conference on computer vision and pattern recognition, pp 2655-2664
25. Zhang G, Luo Z, Cui K, Lu S, Xing EP (2022) Meta-detr: Image-level few-shot detection with inter-class correlation exploitation. IEEE Trans Pattern Anal Mach Intell 45(11):12832–12843
26. Kingma D P (2013) Auto-encoding variational bayes. arXiv:1312.6114
27. Cai B, Yang S, Gao L, Xiang Y (2023) Hybrid variational autoencoder for time series forecasting. Knowl-Based Syst 281
28. Liu W, Li R, Zheng M, Karanam S, Wu Z, Bhanu B, Radke R, Camps O (2020) Towards visually explaining variational autoencoders. In: Proceedings of the IEEE/CVF international conference on computer vision and pattern recognition, pp 8642-8651
29. Zhang J, Bai J, Lin C, Wang Y, Rong W (2022) Improving variational autoencoders with density gap-based regularization. In: Advances in neural information processing systems, pp 19470-19483
30. Baykal G, Kandemir M, Unal G (2024) EdVAE: Mitigating codebook collapse with evidential discrete variational autoencoders. Pattern Recognit 156
31. Chen P, Chen G, Zhang S (2019) Log hyperbolic cosine loss improves variational auto-encoder. IEEE Trans Syst 51(4):2512–24
32. Naesseth C, Ruiz F, Linderman S, Blei D (2017) Reparameterization gradients through acceptance-rejection sampling algorithms. In: Proceedings of the international conference on artificial intelligence and statistics, pp 489-498
33. Tomczak J, Welling M (2018) VAE with a VampPrior. In: Proceedings of the international conference on artificial intelligence and statistics, pp 1214-1223
34. Wang Y, Sun G, Q, (2020) Imbalanced sample fault diagnosis of rotating machinery using conditional variational auto-encoder generative adversarial network. Appl Soft Comput 92
35. Bai R, Huang R, Qin Y, Chen Y, Lin C (2023) HVAE: A deep generative model via hierarchical variational auto-encoder for multi-view document modeling. Inf Sci 623:40–55
36. Higgins I, Matthey L, Pal A, Burgess C, Glorot X, Botvinick M, Mohamed S, Lerchner A (2017) Beta-vae: Learning basic visual concepts with a constrained variational framework. In: Proceedings of the international conference on learning representations, pp 60-79
37. Kumar A, Sattigeri P, Balakrishnan A (2017) Variational inference of disentangled latent concepts from unlabeled observations. arXiv:1711.00848
38. Kolouri S, Pope P E, Martin C E, Rohde G K (2018) Sliced-wasserstein autoencoder: An embarrassingly simple generative model. arXiv:1804.01947
39. Pineau E, Lelarge M (2018) InfoCatVAE: Representation learning with categorical variational autoencoders. arXiv:1806.08240
40. Cai L, Gao H, Ji S (2019) Multi-stage variational auto-encoders for coarse-to-fine image generation. In: Proceedings of the international conference on data mining, pp 630-638
41. Ren S, He K, Girshick R, Sun J (2016) Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39(6):1137–1149
42. Du J, Zhang S, Chen Q, Le H, Sun Y, Ni Y, Wang J, He B, Wang J (2023) $\sigma$-adaptive decoupled prototype for few-shot object detection. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 1578-1587
43. Liu F, Yang S, Chen D, Huang H, Zhou J (2024) Few-shot classification guided by generalization error bound. Pattern Recognit 145
44. Li J, Zhang Y, Qiang W, Si L, Jiao C, Hu X, Zheng C, Sun F (2023) Disentangle and remerge: Interventional knowledge distillation for few-shot object detection from a conditional causal perspective. In: Proceedings of the international conference on artificial intelligence, pp 1323-1333
45. Sun B, Li B, Cai S, Yuan Y, Zhang C (2021) FSCE: Few-shot object detection via contrastive proposal encoding. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7352-7362
46. Yang Z, Wang Y, Chen X, Liu J, Qiao Y (2020) Context-transformer: Tackling object confusion for few-shot detection. In: Proceedings of the AAAI conference on artificial intelligence, pp 12653-12660
47. Yin L, Perez-Rua J, Liang K (2022) Sylph: A hypernetwork framework for incremental few-shot object detection. In: Proceedings of the IEEE/CVF international conference on computer vision and pattern recognition, pp 9001-9010
48. Mpampis E, Passalis N, Tefas A (2023) Symmetric fine-tuning for improving few-shot object detection. In: Proceedings of the symposium series on computational intelligence, pp 598-602
49. Fan Z, Ma Y, Li Z, Sun J (2021) Generalized few-shot object detection without forgetting. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4527-4536
50. Hao P, Wanqi W, Long C, Xianrong P, Jianlin Z, Zhiyong X, Yuxing W, Meihui L (2023) Few-shot object detection via online inferential calibration. Opto-Electron Eng 50(1): 220180-1-220180-14
51. Luo M, Xu J, Fan Y, Zhang J (2022) TRNet: A cross-component few-shot mechanical fault diagnosis. IEEE Trans Indust Inf 19(5):6883–6894
52. Shan D, Zhang Y, Coleman S, Kerr D, Liu S, Hu Z (2022) Unseen-material few-shot defect segmentation with optimal bilateral feature transport network. IEEE Trans Indust Inf 19(7):8072–8082
53. Fan Qi, Zhuo W, Tang C, Tai Y (2020) Few-shot object detection with attention-RPN and multi-relation detector. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4013-4022
54. Xiao Y, Lepetit V, Marlet R (2022) Few-shot object detection and viewpoint estimation for objects in the wild. IEEE Trans Pattern Anal Mach Intell 45(3):3090–3106
55. Kang B, Liu Z, Wang X, Yu F, Feng J, Darrell T (2019) Few-shot object detection via feature reweighting. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 8420-8429
56. He Y, Song K, Meng Q, Yan Y (2019) An end-to-end steel surface defect detection approach via fusing multiple hierarchical features. IEEE Trans Instrum Meas 69(4):1493–1504
57. Ding R, Dai L, Li G, Liu H (2019) TDD-net: a tiny defect detection network for printed circuit boards. CAAI Transactions on Intelligence Technology 4(2):110–116
58. Huang Y, Qiu C, Guo Y, Wang X, Yuan K (2018) Surface defect saliency of magnetic tile. IEEE 14th International Conference on Automation Science and Engineering (CASE), pp. 612-617
59. Huang C, Guan H, Jiang A, Zhang Y, Spratling M, Wang Y (2021) Registration based few-shot anomaly detection. In: Proceedings of the european conference on computer vision, pp 303-319

## Authors and Affiliations

**Ting Zhang[1] · Tianyang You[1] · Zhaoying Liu[1] · Sadaqat Ur Rehman[2] · Yanan Shi[3,4] · Amr Munshi[5]**

✉ Zhaoying Liu
zhaoying.liu@bjut.edu.cn

Ting Zhang
zhangting@bjut.edu.cn

Tianyang You
youtianyang@emails.bjut.edu.cn

Sadaqat Ur Rehman
s.rehman15@salford.ac.uk

Yanan Shi
shiyanan@xjaic.gov.cn

Amr Munshi
aaamunshi@uqu.edu.sa

1 School of Computer Science, Beijing University of Technology, Pingleyuan 100, 100024 Beijing, Beijing, China

2 School of Sciences, Engineering and Environment, University of Salford, M5 4WT Manchester, UK

3 Research Information Department, Xinjiang Uygur Autonomous Region Inspection Institute of Special Equipment, No.188 Hebei East Road, 830011 Urumqi, Xinjiang Uyghur Autonomous Region, China

4 Xinjiang Uyghur Autonomous Region Quality Basic Development Research Institute, No.188 Hebei East Road, 830011 Urumqi, Xinjiang Uyghur Autonomous Region, China

5 Department of Computer and Network Engineering, College of Computing, Umm Al-Qura University, Makkah, Saudi Arabia