Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A graph-based two-stage classification network for mobile screen defect inspection*

Chaofan ZHOU[1,2], Meiqin LIU[‡3,2,1], Senlin ZHANG[1,2], Ping WEI[3], Badong CHEN[3]

[1]*State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China*

[2]*College of Electrical Engineering, Zhejiang University, Hangzhou 310027, China*

[3]*Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an 710049, China*

E-mail: zhouchaofan@zju.edu.cn; liumeiqin@zju.edu.cn; slzhang@zju.edu.cn;
pingwei@mail.xjtu.edu.cn; chenbd@mail.xjtu.edu.cn

Received Oct. 31, 2022; Revision accepted Nov. 28, 2022; Crosschecked Jan. 16, 2023

Abstract: Defect inspection, also known as defect detection, is significant in mobile screen quality control. There are some challenging issues brought by the characteristics of screen defects, including the following: (1) the problem of interclass similarity and intraclass variation, (2) the difficulty in distinguishing low contrast, tiny-sized, or incomplete defects, and (3) the modeling of category dependencies for multi-label images. To solve these problems, a graph reasoning module, stacked on a classification module, is proposed to expand the feature dimension and improve low-quality image features by exploiting category-wise dependency, image-wise relations, and interactions between them. To further improve the classification performance, the classifier of the classification module is redesigned as a cosine similarity function. With the help of contrastive learning, the classification module can better initialize the category-wise graph of the reasoning module. Experiments on the mobile screen defect dataset show that our two-stage network achieves the following best performances: 97.7% accuracy and 97.3% F-measure. This proves that the proposed approach is effective in industrial applications.

Key words: Graph-based methods; Multi-label classification; Mobile screen defects; Neural networks
https://doi.org/10.1631/FITEE.2200524 **CLC number:** TP391.4

1 Introduction

With the mass production of mobile phone screens, accurate and efficient defect inspection is significant for mobile phone screen industrial processes. Manual inspection requires plenty of experienced inspectors, which is labor-consuming and inefficient. In addition, due to the special optical properties of mobile screens, long-term inspection does some harm to the eyes of inspectors. With the rapid development of image processing and deep learning technologies, many automated optical inspection solutions based on convolutional neural networks (CNNs) have been proposed in the textile industry (Wei et al., 2021), steel industry (Wang YC et al., 2021b), mobile phone screen manufacturing industry (Lei et al., 2018; Yuan et al., 2018; Li et al., 2020; Lu et al., 2020; Wang T et al., 2021; Zhang et al., 2021; Zhu et al., 2022), and so on.

This study focuses on mobile screen defect inspection, and the images are obtained by an image acquisition device, as shown in Fig. 1. The device is composed of a line scan camera, an annular light source, transmission parts, a controller, and a computer. Since the original mobile screen image is too large to process directly, we cut it into patches with

‡ Corresponding author

* Project supported by the National Key Research and Development Program of China (No. 2020AAA0108302) and the Fundamental Research Funds for the Central Universities, China (No. xtr072022001)

ORCID: Chaofan ZHOU, https://orcid.org/0000-0003-0807-6539; Meiqin LIU, https://orcid.org/0000-0003-0693-6574

the size of 1024×1024. The cropped image patches are displayed in Fig. 2.

Although directly applying the CNN model to defect inspection can achieve better results than traditional approaches that use hand-crafted features, there are some challenging issues brought by task-specific characteristics of mobile screen defect inspection. We show some samples of mobile screen defects in Fig. 2. However, the original image is dark and inconvenient to see, as presented in Fig. 2a. Therefore, the rest of the images in Fig. 2 are brightened for easier viewing. First, some defects have the characteristics of interclass similarity and intraclass variation (ISIV). For example, the scratches in Figs. 2b and 2c belong to the same category, but their appearances

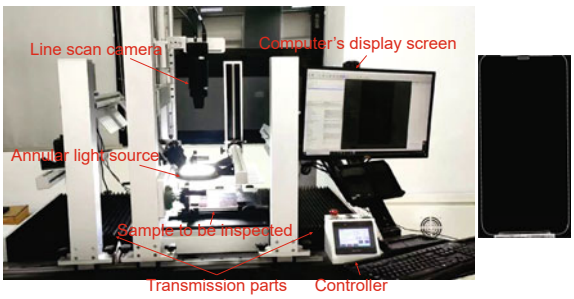

Fig. 1 Image acquisition device and an image obtained by the device

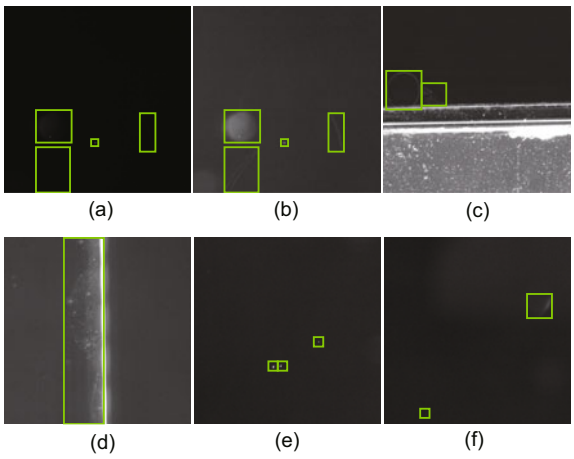

Fig. 2 Samples of mobile screen defects indicated by manual annotation: (a) original image; (b-f) images after adjusting the brightness for view, where defects that appear in the corresponding subfigures from left to right and top to bottom are bubble, scratch, pinhole, scratch (a, b), scratch, tin ash (c), tin ash (d), pinhole, pinhole, pinhole (e), and pinhole, bubble (f). Some defects have the characteristics of interclass similarity and intraclass variation (ISIV). Some defects are of low contrast or are tiny-sized or incomplete

vary greatly. The scratch in Fig. 2c and the bubble in Fig. 2b are somewhat similar in shape. The local appearance of tin ash in Fig. 2d is rather like the appearance of a pinhole, presented in Fig. 2e. As a consequence of this characteristic, it is hard to classify defects correctly. Second, some defects on mobile screen images are of low contrast or are tiny-sized or incomplete. For example, the bubble in Fig. 2f is not as complete as the bubble in Fig. 2b. The pinhole in Fig. 2f is too small and of too low contrast to be seen. These phenomena make it easy to miss defects during inspection. Third, we need to address the problem of multi-label classification. Compared with single-label classification, multi-label classification is a more practical and general problem (Wei et al., 2021), as a real-world image generally contains single or multiple defect classes, and so do the data we collect. As far as we know, there are a few research works (Lei et al., 2018; Li et al., 2020; Lu et al., 2020) on mobile screen defect classification up to now, but they study only single-label classification, which aims at classifying an image into one class. Different from them, we address the problem of multi-label classification on a mobile screen defect dataset. The multi-label task is more challenging, and a key to multi-label classification is to model the category dependencies.

To address the above issues, a two-stage defect inspection network is proposed for multi-label classification of mobile phone screen defects. The proposed network consists of two components, i.e., a feature-clustering-based classification module (FC2-module) and a graph reasoning module (GR-module). To address the first problem, we create a category-to-category graph in the GR-module. The graph enhances the image features with the help of category features to increase the interclass distance and the intraclass correlation in the feature space.

To obtain better node representations of the graph in the GR-module, we first use contrastive learning in the FC2-module to pull together features with the same label, while pushing apart features with different labels. Then, multiple category clusters are formed. However, contrastive learning cannot be directly applied to the multi-label problem; so, we propose the category pooling operator to generate label-level features. The classifier of the FC2-module is designed as a cosine similarity function to make the weights represent the overall features. We

use the distilling weights as node representations.

Besides, in the GR-module, we establish image-wise relations to improve features of images that have low contrast, tiny-sized, or incomplete defects. The use of image-wise relations is inspired by this phenomenon. When humans are not sure about the objects in the image being viewed, they refer to other images in which similar objects occur. Moreover, we model the category dependencies using the graph convolution network (GCN) and the co-occurrence patterns of defects.

In summary, the major contributions of this paper are as follows:

1. Based on a category graph and image relations, the GR-module exploits category-wise dependency, image-wise relations, and interactions between them to enhance original image features. The category correlation introduced in the image features is applied to increase the interclass distance and decrease the intraclass distance in the feature space. With image-wise relations, the features of low-quality images can be improved with reference to other high-quality images. Category dependencies embodied in the category graph help improve the multi-label classification performance.

2. The FC2-module is designed to help the GR-module better address the three problems mentioned above. For the better initialization of the graph of the GR-module, and consequently improving the multi-label classification performance, we propose a novel classification module (i.e., FC2-module). This module uses contrastive learning and cosine similarity to obtain weights that contain enough category information for initialization. We design a simple category pooling operator to solve the dilemma of applying contrastive learning to multi-label defect inspection.

3. Our experiments on a real-world industrial dataset demonstrate that the proposed two-stage classification network is effective in mobile screen defect inspection.

2 Related works

2.1 Defect inspection

This subsection introduces mainly approaches for mobile screen defect inspection, graph-based defect classification, and multi-label defect classification. Nowadays, many CNN-based models have been widely used in classification (Lei et al., 2018; Li et al., 2020; Lu et al., 2020), object detection (Wang T et al., 2021; Zhang et al., 2021; Zhu et al., 2022), and segmentation (Yuan et al., 2018) for mobile screen defect inspection. These works demonstrate that CNNs can capture much richer features to overcome the limitations of traditional methods, i.e., the methods applying hand-crafted features. However, different from our work, some approaches (Lei et al., 2018; Li et al., 2020; Lu et al., 2020) are aimed at solving the problem of single-label classification, while others (Yuan et al., 2018; Wang T et al., 2021; Zhang et al., 2021; Zhu et al., 2022) experiment only on single-label images. However, the images we face are multi-label. A key to multi-label classification is modeling category dependencies, which is not considered by these works. Besides, because only one or two defects are considered, or the task type is different, they do not research ISIV, the factor that is prone to cause wrong inspection.

Some studies (Wang YC et al., 2021a, 2021b; Kong et al., 2022; Xiao et al., 2022) introduced the GCN for surface defect classification. A semi-supervised algorithm was proposed by Wang YC et al. (2021a). They used the GCN to propagate label information so that unlabeled images could obtain labels. Wang YC et al. (2021b) applied the GCN for defect classification and addressed the problem of class imbalance. Xiao et al. (2022) addressed the few-shot problem and introduced the graph to embed features of the support set into query features. Different from us, the graphs proposed by these authors (Wang YC et al., 2021a, 2021b; Xiao et al., 2022) were image-to-image graphs and were applicable to only single-label classification, not multi-label classification. Kong et al. (2022) designed a knowledge graph but considered only the knowledge between bolt and nut pairs. They did not consider the relationship between the same categories.

Some works (Park et al., 2020; Haurum and Moeslund, 2021; Wei et al., 2021) do address the problem of multi-label defect classification. Nevertheless, the field and characteristic of defects they studied are different from those of ours. They studied defects in printed circuit board (PCB) screen printer, sewer, and textile, respectively. Directly applying these methods to mobile screen defects may degrade the classification performance. As shown in

Section 4.3, the model (Haurum and Moeslund, 2021) does not perform well on our dataset. Based on the characteristics of mobile screen defects, we propose a different method that exploits category-wise dependency, image-wise relations, and interactions between them.

2.2 Graph-based multi-label classification

Modeling the label dependencies is a key to help multi-label classification. There are quite a few works attempting to capture category dependencies by using probabilistic graph model, attention mechanism, GCN (Chen ZM et al., 2019; Wang Y et al., 2020; Zhao et al., 2021), and so on. Graph has been proved to be more effective in modeling category correlation (Chen ZM et al., 2019). Although these GCN-based methods have achieved appealing performance in natural image tasks, it is unwise to use them directly in industrial images. Some methods (Chen ZM et al., 2019; Wang Y et al., 2020) initialize graph nodes with linguistic word embeddings learned by language representation models. They ignore the characteristics of each image. Besides, training word embeddings for classifying industrial defects is hard. Different from them, we use a dynamically updated graph, whose node representations are initialized by the weights of the classifier. Additionally, other methods (Chen ZM et al., 2019; Wang Y et al., 2020; Zhao et al., 2021) neglect image-wise relations and image-to-category interactions, which we use to improve the classification performance.

2.3 Contrastive learning

Contrastive learning is proposed to pull together features with the same label while pushing apart features with different labels under self-supervised or fully-supervised settings. Under the fully-supervised setting, ground-truth label information is leveraged. Under the self-supervised setting, features generated from different images are deemed to have different labels. Recent works (Wu et al., 2018; Hjelm et al., 2019; Chen T et al., 2020; He et al., 2020; Khosla et al., 2020) have shown that contrastive learning can learn better representations and help improve performance in classification tasks. Contrastive learning under the self-supervised setting (Wu et al., 2018; Hjelm et al., 2019; Chen T et al., 2020; He et al., 2020) can work without labels, thereby relieving the burden of manual annotations. Nonetheless, for an image, these works (Wu et al., 2018; Hjelm et al., 2019; Chen T et al., 2020; He et al., 2020) took only augmented versions of the same image as positives, while methods under the fully-supervised setting (Khosla et al., 2020) also took images of the same class as positives. This principle causes the performance of some methods (Wu et al., 2018; Hjelm et al., 2019; Chen T et al., 2020; He et al., 2020) to be generally lower than the supervised contrastive method (Khosla et al., 2020). Consequently, in this study, the supervised contrastive method (Khosla et al., 2020) is introduced to learn better representations for our graph nodes. However, due to the multi-label attribute of our dataset, image features cannot be directly input to the supervised contrastive method (Khosla et al., 2020). To address this shortcoming, we introduce a new simple method to produce label-level features of an image. Because the dimension of label-level features is low, the projection network in the original implementation is not required.

3 Two-stage network and training policy

In this section, we describe our two-stage network for mobile screen defect inspection and two-phase training policy in detail.

As depicted in Fig. 3, the proposed two-stage network consists of two components, i.e., the FC2-module and the GR-module. The former is able to classify defects and supply a good initialization for the graph of the latter, while the latter can be stacked on any existing classification network and improves the classification performance by reasoning semantics over image features with a category-wise graph.

Given a batch of input images $\boldsymbol{I} \in \mathbb{R}^{B\times W\times H\times 3}$, the FC2-module first uses a feature extractor and the category pooling operator to extract $C \times d$-dimensional features $\boldsymbol{X}$ as follows:

$$\boldsymbol{X} = \text{pooling}\big(F(\boldsymbol{I})\big) \in \mathbb{R}^{B\times Cd}, \tag{1}$$

where B is the batch size, H and W denote the height and width of the images respectively, and C means the number of categories. The pooling$(\cdot)$, referring to the category pooling operator, consists of a global average pooling layer and a linear layer. The features are fed into the first classifier to obtain classification scores $\boldsymbol{S} = \text{Cls}_1(\boldsymbol{X}|\boldsymbol{w}_{\text{c1}}) \in \mathbb{R}^{B\times C}$ of B images ($\boldsymbol{w}_{\text{c1}}$

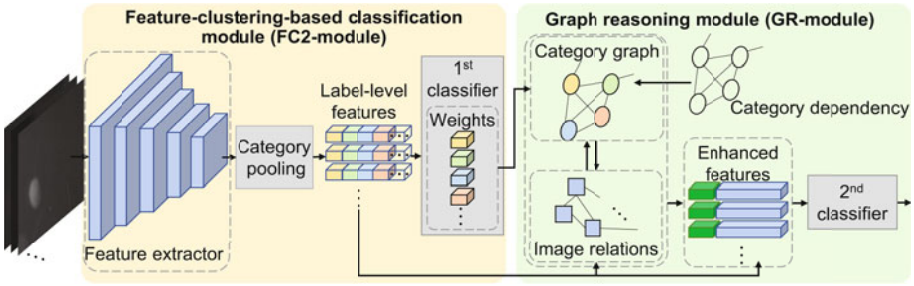

Fig. 3 The proposed two-stage network. The input images are sent to the feature-clustering-based classification module (FC2-module) to obtain label-level features and distilling weights. Features and weights are fed into the graph reasoning module (GR-module) to enhance features. Finally, enhanced features are used to improve the classification performance

are learnable parameters of Cls_1). Then, the features $\boldsymbol{X}$ are enhanced by reasoning semantics over image features with a category-to-category undirected graph G. Finally, the enhanced features $\boldsymbol{X}'$ are sent to the second classifier to obtain the final classification scores $\boldsymbol{S}' = \mathrm{Cls}_2(\boldsymbol{X}'|\boldsymbol{w}_{\mathrm{c2}}) \in \mathbb{R}^{B\times C}$ ($\boldsymbol{w}_{\mathrm{c2}}$ are learnable parameters of Cls_2). To obtain better node features of G, we implement the first classifier as a cosine similarity function between feature vectors and classification vectors, which is explained in detail in Section 3.1.

3.1 FC2-module

To make graph G better promote the model's performance, we use the distilling weights of the first classifier as the representations of category nodes. To enhance the category representation capability of weights, the FC2-module is designed to contain three essential blocks: (1) a category pooling operator that changes image-level features to label-level features; (2) contrastive learning, improved based on multi-label characteristics, which gathers label-level features with the same label; (3) a classifier that makes the weights represent the overall features of each cluster. Next, the structure of this FC2-module is introduced in detail.

Recently, the resurgence of the study of contrastive learning has brought about major advances in classification by learning better representations. To enhance the category representation capability of the weights of the first classifier, we add supervised contrastive loss to the FC2-module. Our dataset belongs to a multi-label dataset. Different from single-label classification based on contrastive learning (Khosla et al., 2020), multi-label classification is more complex. Image-level features extracted by the multi-label classification module cannot be used directly as label-level features. Hence, it is hard to define positive or negative samples for an anchor. We solve this problem by designing the category pooling operator. Taking the three-dimensional (3D) feature of an image as input, the category pooling operator uses a global average pooling layer to aggregate spatial information, and then applies a linear layer of size $C\times d$ to obtain features $\boldsymbol{X}\in\mathbb{R}^{Cd}$, which can be seen as a set of label-level features $\left\{\boldsymbol{X}^i\in\mathbb{R}^d\right\}_{i=1}^{C}$.

The pipeline of updating the weights of the first classifier is depicted in Fig. 4. Given a batch of input images, all label-level features can be seen as $\boldsymbol{X} = \left\{\boldsymbol{X}^{ij}\in\mathbb{R}^d|i\in\{1,2,...,B\};j\in\{1,2,...,C\}\right\}$ and ground-truth labels can be expressed as $Y = \left\{Y^{ij}\in\{0,1\}|i\in\{1,2,...,B\};j\in\{1,2,...,C\}\right\}$.

We define $A_{\mathrm{I}} = \{\boldsymbol{X}^{ij}\in\boldsymbol{X}|Y^{ij}=1\}$ as a set that contains label-level features with active labels, and $A_{\mathrm{D}}(i,j) = A_{\mathrm{I}}\backslash\{\boldsymbol{X}^{ij}\}$ as a set that contains all label-level features in A_{I} except $\boldsymbol{X}^{ij}$. We define $\boldsymbol{X}^{ij}$ as the anchor. The positive set can be expressed as $A_{\mathrm{P}}(i,j) = \left\{\boldsymbol{X}^{\mathrm{p}j}\in A_{\mathrm{D}}(i,j)|Y^{\mathrm{p}j}=Y^{ij}=1\right\}$, which contains label-level features with the same active labels as $\boldsymbol{X}^{ij}$ except $\boldsymbol{X}^{ij}$. The negative set can be defined as $A_{\mathrm{N}}(i,j) = A_{\mathrm{D}}(i,j)\backslash A_{\mathrm{P}}(i,j)$, which contains label-level features with different active labels. With the above notations, the supervised contrastive loss we use is written as shown in Eq. (2). In Eq. (2), τ is a scalar temperature parameter and dist$(\cdot)$ refers to the distance function of contrastive loss. Previous research works (Gidaris and Komodakis, 2018; Khosla et al., 2020) have proved that cosine similarity space can obtain better classification performance than just linear combination. Therefore, we apply the cosine similarity as the distance function

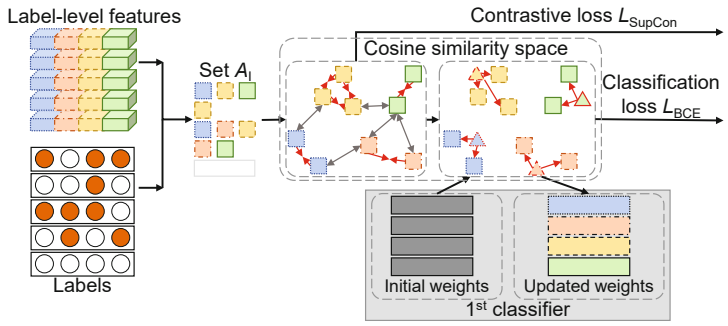

Fig. 4 The pipeline of updating the weights of the first classifier. This figure takes as an example the condition when the batch size is five and the number of categories is four. Four different types of borders in the features represent four different categories. The triangle in the cosine similarity space represents the weight of the classifier

$$L_{\mathrm{SupCon}}=\frac{1}{|A_{\mathrm{I}}|}\sum_{\boldsymbol{X}^{ij}\in A_{\mathrm{I}}}\frac{-1}{|A_{\mathrm{P}}(i,j)|}\sum_{\boldsymbol{X}^{\mathrm{P}}\in A_{\mathrm{P}}(i,j)}\log\frac{\exp\big(\mathrm{dist}(\boldsymbol{X}^{ij},\boldsymbol{X}^{\mathrm{P}})/\tau\big)}{\sum_{\boldsymbol{X}^{a}\in A_{\mathrm{D}}(i,j)}\exp\big(\mathrm{dist}(\boldsymbol{X}^{ij},\boldsymbol{X}^{a})/\tau\big)}. \tag{2}$$

of contrastive loss to form multiple category clusters. Furthermore, to make the weights of the classifier learn enough information about each defect category, we implement the first classifier as a cosine similarity function between feature vectors and weight vectors. Different from the conventional linear classifier, this classifier applies label-level features rather than image-level features as input and places features in the normalized space (i.e., cosine similarity space) before outputting the classification scores $\boldsymbol{S}$.

We apply binary cross-entropy (BCE) loss as the classification loss. In the cosine similarity space, the gradient of classification loss tends to encourage the weights of the first classifier to learn the overall features of each defect category. Finally, we obtain the updated weights that contain enough information about each defect category.

3.2 GR-module

3.2.1 Graph construction

The GR-module is based on a category-to-category undirected graph $G=(\boldsymbol{N},\boldsymbol{E})$, where $\boldsymbol{N}\in\mathbb{R}^{C\times d}$ refers to a set of category nodes and $\boldsymbol{E}\in\mathbb{R}^{C\times C}$ is a set of edges encoding relations between corresponding categories. Inspired by current works (Gidaris and Komodakis, 2018; Xu et al., 2019), in which the authors thought that the weights of the classifier actually contain high-level semantic information for each category, we use the distilling weights of the first classifier to be the representations of category nodes. The edges of graph G refer to co-occurrence between corresponding categories. To construct $\boldsymbol{E}$, we first count the occurrence of label pairs in the training set and obtain the statistical matrix $\boldsymbol{M}\in\mathbb{R}^{C\times C}$. Then, row normalization is performed to obtain the adjacent matrix $\boldsymbol{E}$: $E_{ij}=\frac{M_{ij}}{\sum_{k=1}^{C}M_{ik}}$. Finally, we set $\boldsymbol{E}$ as learnable parameters, which can be adaptively adjusted through iterations.

3.2.2 Pipeline of the GR-module

Based on graph G, this module exploits category-wise dependency, image-wise relations, and interactions between them. By doing so, the GR-module can incorporate category semantics and image relations to guide enhancement of image features. Here, we explain the pipeline of the GR-module in detail.

First, the GCN is instantiated and category representations are updated according to the prior knowledge graph edges $\boldsymbol{E}$, which is formulated as follows:

$$\boldsymbol{N}'=\hat{\boldsymbol{E}}\boldsymbol{N}, \tag{3}$$

where $\hat{\boldsymbol{E}}$ is the normalized version of $\boldsymbol{E}$, i.e., $\hat{\boldsymbol{E}}=\boldsymbol{D}^{-1/2}\boldsymbol{E}\boldsymbol{D}^{-1/2}$. Here, $\boldsymbol{D}$ is diagonal, and $D_{ii}=\sum_j E_{ij}$.

Second, image-wise relations are modeled. This process can be described by the following two formulae:

$$\boldsymbol{Z}=\mathrm{Proj}(\boldsymbol{X}), \tag{4}$$

$$R_{ij}=\frac{\exp(\boldsymbol{Z}_i\boldsymbol{Z}_j^{\mathrm{T}})}{\sum_{k=1}^{B}\exp(\boldsymbol{Z}_i\boldsymbol{Z}_k^{\mathrm{T}})}, \tag{5}$$

where $\mathrm{Proj}(\cdot)$ is a projection function and $i=1,2,...,B, j=1,2,...,B$. In mobile screen images, some defects are of low contrast, tiny-sized, or incomplete. Detecting these defects is challenging because they have low resolution and limited information. The output resolution of CNN is usually much lower than the input resolution. This usually leads to lower resolution and lower discrimination of features corresponding to these defective regions. Therefore, the features of these images are not enough to represent some categories. This leads to the low scores of these categories. Furthermore, misclassification occurs. We hope that low-quality image features can be improved by exploiting image-wise relations. The application of image-wise relations is inspired by this phenomenon. When humans are not sure about the objects in the image being viewed, they refer to other images in which similar objects occur. Specifically, as in Eq. (4), we apply the projection function $\mathrm{Proj}(\cdot)$ to map image features to a latent space for similarity computation. The projection function can also condense channels to reduce computation complexity. We instantiate the projection function $\mathrm{Proj}(\cdot)$ as just a single linear layer. As in Eq. (5), we apply the dot-product operator and softmax operator to perform the similarity computation. Finally, the relations of images in the same batch are obtained.

Third, the mapping between images and categories is established with the help of other images, formulated as follows:

$$\boldsymbol{R}'=\boldsymbol{R}\,\mathrm{Soft}(\boldsymbol{S}), \tag{6}$$

where $\mathrm{Soft}(\cdot)$ is a softmax function. This function is used to normalize the scores $\boldsymbol{S}$ over all categories.

Next, new image features are generated under the guidance of category semantics and image relations. We concatenate the original image features $\boldsymbol{X}$ with the new image features to produce the enhanced features $\boldsymbol{X}'$. The process can be formulated as follows:

$$\boldsymbol{X}'=\mathrm{Concat}\big(\boldsymbol{X},F_{\mathrm{a}}(\boldsymbol{R}'\boldsymbol{N}'\boldsymbol{w}_{\mathrm{RN}})\big), \tag{7}$$

where $\boldsymbol{w}_{\mathrm{RN}}$ denotes the set of learnable parameters, $F_{\mathrm{a}}(\cdot)$ is an activation function, and $\mathrm{Concat}(\cdot)$ refers to a concatenation function. The enhanced features $\boldsymbol{X}'$ have category relevance in the latter dimensions. This category correlation increases the interclass distance and decreases the intraclass distance in the feature space.

Finally, the enhanced features $\boldsymbol{X}'$ are fed into the second classifier to obtain better classification results. The second classifier is implemented as a single linear layer without bias. We select BCE loss as the classification loss of the second classifier.

3.3 Two-phase training policy

Node information has an effect on the initial point of the model in the optimization space. It plays an important role in reinforcing the learning capabilities of the GR-module. However, during the initial training, the weights of the first classifier, which are used as representations of $\boldsymbol{N}$, are initialized randomly. The weights cannot contain high-level semantic information to express category representations. We can alleviate this problem by adjusting the proportion of the loss of the second classifier in the total loss. This can be implemented in many ways, including (1) progressively tuning up the proportion in predefined milestones and (2) designing a function for changing the proportion. However, experiments prove that they are time-consuming and hard to debug. Hence, we ultimately choose to pretrain the first classifier to obtain weights that contain enough category information. This is more efficient and easier to debug. In summary, the training procedure is split into two phases: the node pretraining phase and the normal training phase. We use the same training set as the input for both phases.

In the first phase, we train only the FC2-module with classification loss L_{BCE} and contrastive loss L_{SupCon}. The parameters of the feature extractor and the weights of the first classifier are learned. The overall training loss of this phase can be expressed as follows:

$$L_{\mathrm{Pre}}=L_{\mathrm{BCE}}+\alpha L_{\mathrm{SupCon}}, \tag{8}$$

where the scalar parameter α controls the trade-off between classification loss and contrastive loss.

After the weights of the first classifier are learned to contain enough high-level semantic information, we apply them to initialize the category nodes of graph G. Then, the training enters the second phase, during which we train the learnable parameters of the GR-module while we continue

training the parameters of the FC2-module. The whole network is subjected to end-to-end training with the average sum of classification loss L_{BCE} of the two classifiers. Algorithm 1 summarizes the whole training process of the proposed network. We empirically discover that the proposed two-phase training policy is effective.

Algorithm 1 The training pipeline of the two-stage network

// The node pretraining phase
Input: Training set D_{train}
Output: A trained FC2-module, initialized category nodes $\boldsymbol{N}$
for epoch $k = 1, 2, \ldots$ **do**
  $\boldsymbol{I} \leftarrow$ Sample the training images from D_{train}
  $\boldsymbol{X} \leftarrow$ Extract the label-level features for all images in $\boldsymbol{I}$ with Eq. (1)
  $L_{\mathrm{SupCon}} \leftarrow$ Compute the contrastive loss by applying Eq. (2) on features $\boldsymbol{X}$
  $\boldsymbol{S} \leftarrow$ Compute classification scores for the label-level features $\boldsymbol{X}$ with the first classifier
  $L_{\mathrm{BCE}} \leftarrow$ Compute the classification loss with scores $\boldsymbol{S}$
  Optimize the parameters of the FC2-module with Eq. (8), which is a weighted combination of L_{BCE} and L_{SupCon}
end for
Initialize category nodes $\boldsymbol{N}$ of graph G with the weights of the first classifier
Construct edges $\boldsymbol{E}$ of graph G with labels of D_{train}
// The normal training phase
Input: Training set D_{train}, pretrained FC2-module, initialized graph G
Output: The trained whole model, including the FC2-module and the GR-module
for epoch $k = 1, 2, \ldots$ **do**
  $\boldsymbol{I} \leftarrow$ Sample the training images from D_{train}
  $\boldsymbol{X} \leftarrow$ Extract the label-level features for all images in $\boldsymbol{I}$ with Eq. (1)
  $\boldsymbol{X}' \leftarrow$ Use $\boldsymbol{X}$ to produce the enhanced features $\boldsymbol{X}'$ with Eqs. (3)–(7)
  $L \leftarrow$ Compute the average sum of classification losses (i.e., BCE losses) of the two classifiers
  Optimize the parameters of the FC2-module and the GR-module with L
end for

4 Experimental results and discussion

4.1 Dataset and evaluation metrics

We collect 80 mobile screen defect images with a size of $5120 \times 10\,240$ from real industrial scenarios, where 56 images are for training, 12 images are for validation, and 12 images are for testing. We cut patches with the size of 1024×1024 from the original images with a sliding window. The categories of patches are annotated by experienced inspectors. The patches whose defective area accounts for $\leq 20\%$ of the original defective area are not considered. The types of mobile screen defects include bubble, pinhole, scratch, and tin ash. Each image patch contains 0–4 types of defects. The number of patches of each category before applying data augmentation can be seen in Table 1.

Four metrics are adopted to evaluate the classification performance: accuracy, precision, recall, and F-measure. They are broadly used to evaluate the performance of the multi-label classification models. For each image, the predicted labels are deemed as positive when their confidence levels are >0.5. Of the four metrics, accuracy and F-measure are applied to evaluate the overall performance.

4.2 Implementation details

Our approach is implemented using the PyTorch (Paszke et al., 2017) framework on a server with Ubuntu 16.04 operating system and four NVIDIA TITAN X graphics processing units (GPUs). ResNet-50 (He et al., 2016) is used to implement the feature extractor $F(\cdot)$. During pretraining and normal training, the stochastic gradient descent (Bottou, 2010) is used with a momentum of 0.9, a weight decay parameter of 0.0005, and a batch size of 32. We set the initial learning rate of both phases as 0.001. All iteration training is terminated after 40 epochs. The dimension of label-level features d is set to 512. Following Khosla et al. (2020), the temperature parameter τ in Eq. (2) is 0.1. Moreover, α in Eq. (8) is set to 0.4. Data augmentation is commonly used to increase the diversity of the dataset, avoid overfitting of complex networks, and enhance the robustness of networks. We apply some data augmentations to the training set, including horizontal image flipping, vertical image flipping, rotation, and color jitter. The size of the training image is set as 1024×1024.

4.3 Performance evaluation

We compare our approach with state-of-the-art methods, including three prevalent CNN classification methods (ResNet-50 (He et al., 2016), VGG-16 (Simonyan and Zisserman, 2015), and Inception-v3 (Szegedy et al., 2016)), two graph-based

Table 1 Number of patches of each category in the training, validation, and testing sets

Category	Number of patches in the training set	Number of patches in the validation set	Number of patches in the testing set
Bubble	91	21	33
Pinhole	1514	229	344
Scratch	1010	235	228
Tin ash	726	194	130
Defect-free	540	123	111
Total patches	2520	503	578

multi-label classification methods (KSSNet (Wang Y et al., 2020) and TDRG (transformer-based dual relation graph) (Zhao et al., 2021)), and one multi-label defect inspection method (SewerMl (Haurum and Moeslund, 2021)). Both KSSNet and TDRG construct category-wise relations using GCN to improve the performance of multi-label classification. The SewerMl mentioned by Haurum and Moeslund (2021) is a multi-label classification framework for defect inspection.

Table 2 gives the results of the multi-label classification on the testing set. These metrics of classification performance are averaged over categories. It shows that our approach achieves the highest accuracy (97.7%), precision (97.5%), recall (97.0%), and F-measure (97.3%) on the mobile screen defect dataset. Compared with the second place of each metric, our model outperforms with a margin of 0.5% on accuracy, 0.9% on precision, 1.1% on recall, and 1.2% on F-measure. These results prove that the proposed method is feasible. TDRG and Inception-v3 also achieve relatively good results. TDRG designs a category-wise graph and dynamically constructs the correlation matrix as graph edges in a learnable manner. This illustrates that exploring category relations based on the graph does help multi-label classification of mobile phone screen defects. Inception-v3 has a relatively large number of network branches, which allows the final extracted features to have different receptive fields. This may help it better identify defects of different sizes on the same image.

Table 2 Comparisons with state-of-the-art methods on a multi-label screen defect classification task

Method	Accuracy	Precision	Recall	F-measure
ResNet-50	0.952	0.966	0.919	0.942
VGG-16	0.967	0.963	0.827	0.890
Inception-v3	0.971	0.963	0.953	0.958
KSSNet	0.953	0.828	0.713	0.767
TDRG	0.972	0.963	0.959	0.961
SewerMl	0.887	0.820	0.727	0.771
Ours	**0.977**	**0.975**	**0.970**	**0.973**

The results in bold are the highest

The per-class precision and recall of our approach are shown in Fig. 5. Compared with other types of defects, the precision and recall for the defect "pinhole" are the lowest, which are 95.9% and 94.5%, respectively. The inspection performance of each category basically meets the needs of actual screen defect inspection.

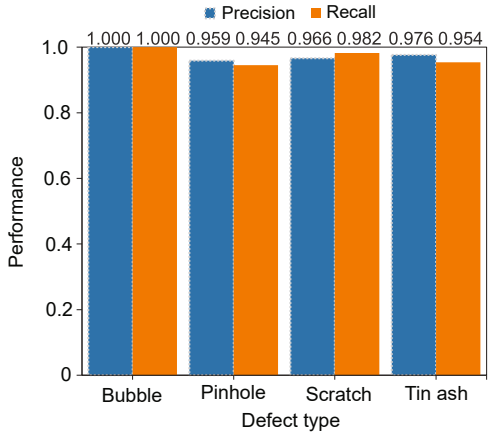

Fig. 5 Per-class precision and recall of our approach on the mobile screen defect dataset

4.4 Ablation studies

4.4.1 Effects of training policy and components

We perform ablation studies on the mobile screen defect dataset to verify the effectiveness of the GR-module, the node pretraining phase, and the cosine similarity space of the proposed approach. As illustrated in Table 3, the GR-module helps the proposed approach by reasoning semantics over image features with a category-to-category undirected graph. It increases accuracy by 0.8% and F-measure

by 0.6%. The node pretraining phase can improve performance by 0.4% in terms of accuracy and 0.8% in terms of F-measure. This indicates that it is important for our approach to improve the initial node features of the graph G. The node pretraining phase is beneficial in that point.

To explore how much the cosine similarity space contributes to the improved results compared to the linear combination space, we conduct experiments. The only difference between the two kinds of spaces is that all the features of the former are normalized. Specifically, the cosine similarity space in Table 4 refers to the fact that we apply the cosine similarity as the distance function for contrastive learning, and the first classifier adopts the cosine similarity function. Linear combination space is implemented by using the dot-product distance function and the conventional linear classifier. As illustrated in Table 4, using cosine similarity space can obtain better classification performance than just applying linear combination space.

Overall, the experimental results show that the training policy and the two components of the proposed approach all can improve the performance of the proposed approach.

4.4.2 Parameter analysis

In this subsection, we detail the experiments to show the influence of parameters and to find their most appropriate values.

1. Parameter d of the category pooling operator

The category pooling operator is applied to generate label-level features. The value of d determines the dimension of the label-level features and is critical to the performance of the model. We experiment with different values of d, and the results are shown in Table 5. Compared with other values, when d equals 512, the best results are achieved on almost all metrics. Fig. 6a visualizes intuitively how performance changes with parameter d. We can see that the performance of the model is generally improved with the increase of d at the beginning. When the value of d exceeds 512, the performance of the model begins to deteriorate. According to Table 5 and Fig. 6a, the most appropriate value of d is 512, where the dimension of label-level features is enough to represent useful information without introducing too much noise. This might mean that the dimension of the label-level features is best set close to the dimension of the image-level features divided by the number of classes.

2. Parameter α in Eq. (8)

Parameter α controls the contribution of contrastive learning to the total loss in the first phase. To explore how much contrastive learning actually contributes to the improved results, different values of α are selected for the experiments. The results are presented in Table 6, and the results of accuracy and F-measure are plotted in Fig. 6b. When contrastive learning is not applied or when α is small, the label-level features are hard to cluster by categories, which prevents the weights of the first classifier from

Table 3 Ablation results of the GR-module and the node pretraining phase on the mobile screen defect dataset

GR-module	Pretrain	Accuracy	Precision	Recall	F-measure
✗	✗	0.965	0.964	0.953	0.959
✓	✗	0.973	0.969	0.961	0.965
✓	✓	**0.977**	**0.975**	**0.970**	**0.973**

The results in bold are the highest

Table 4 Ablation results of cosine similarity space on the mobile screen defect dataset

Space	Accuracy	Precision	Recall	F-measure
Linear combination	0.971	0.968	0.957	0.963
Cosine similarity	**0.977**	**0.975**	**0.970**	**0.973**

The results in bold are the highest

Table 5 Ablation results of different values of d on the mobile screen defect dataset

d	Accuracy	Precision	Recall	F-measure
64	0.971	0.968	0.959	0.964
128	0.974	0.965	0.964	0.965
256	0.972	0.976	0.961	0.969
512	**0.977**	0.975	**0.970**	**0.973**
1024	0.973	0.971	0.964	0.968
2048	0.970	**0.978**	0.953	0.965

The results in bold are the highest. d: dimension of the label-level features

Table 6 Ablation results of different values of α on the mobile screen defect dataset

α	Accuracy	Precision	Recall	F-measure
0	0.971	0.962	0.961	0.962
0.2	0.974	**0.977**	0.961	0.969
0.4	**0.977**	0.975	**0.970**	**0.973**
0.6	0.973	0.961	0.966	0.964
0.8	0.972	0.968	0.962	0.965
1.0	0.969	0.958	0.956	0.957

The results in bold are the highest. α: parameter that controls the trade-off between the classification loss and contrastive loss

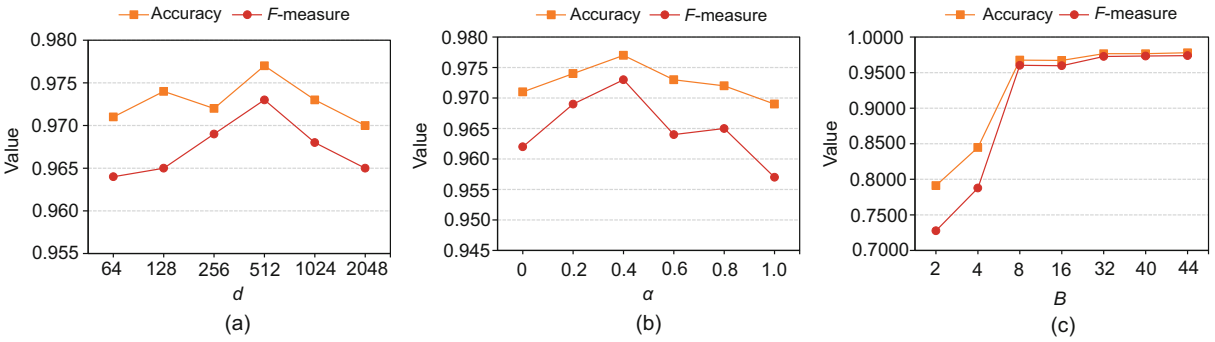

Fig. 6 Trends of accuracy and *F*-measure with different parameters (*d*, α, and *B*) on the mobile screen defect dataset. *d*: dimension of the label-level features; α: parameter that controls the trade-off between the classification loss and contrastive loss; *B*: batch size

learning well. When α is too large, the contribution to classification loss is degraded too much, which prevents the weights from containing sufficient category information. From Table 6, we can see that the model can achieve better performance with the assistance of contrastive learning. The most appropriate value of α is 0.4, which makes the proposed model acquire the best performance on almost all metrics.

3. Batch size B in the second phase

Image relation is an integral part of the loss of the GR-module, and the loss is computed only within batches. Consequently, the batch size B may have effects on the multi-label classification results. In the spirit of fairness, we use the same model parameters pretrained in the first phase and alter the value of B in the second phase. All experimental results are shown in Table 7, and the results of accuracy and F-measure are depicted in Fig. 6c. When B becomes larger, the change trend of performance slows down; so, four decimal places are reserved in Table 7 and Fig. 6c to present the change more clearly. We can observe that the performance of the proposed network is basically enhanced with the increase of B. However, for the fairness of experiments, the batch size B of all models is set to 32.

4.5 Method complexity

Parameter counts and the number of FLoating-point OPerations (FLOPs) of different models with input image size of 1024×1024 are reported in Table 8. The former metric reflects the memory size of the model, and the "FLOPs" is used to measure the complexity of the model. Our proposed model has 2.469×10^7 parameters and has about 8.610×10^{10} FLOPs. Models that perform better, namely, ResNet-50, Inception-v3, TDRG, and our proposed model, have similar model complexity. Our proposed network needs about 4.5 h to train on the mobile screen defect dataset. In the real-world phone screen production line, the original image to be inspected is cropped into multiple patches with a sliding window. The image patches are detected in parallel with our proposed network. The results of the same original image are summarized to achieve classification and coarse location of defects. The proposed network is able to achieve the speed of 156 frames/s on image

Table 7 Ablation results of different values of *B* on the mobile screen defect dataset

B	Accuracy	Precision	Recall	*F*-measure
2	0.7911	0.6372	0.8479	0.7276
4	0.8447	0.7417	0.8398	0.7877
8	0.9676	0.9647	0.9560	0.9603
16	0.9671	0.9638	0.9556	0.9597
32	0.9766	0.9752	0.9703	0.9727
40	0.9766	**0.9826**	0.9641	0.9733
44	**0.9779**	0.9769	**0.9710**	**0.9739**

The results in bold are the highest. *B*: batch size

Table 8 Comparison of numbers of parameters and FLOPs of different methods with input image size of 1024×1024

Method	Number of parameters ($\times 10^7$)	Number of FLOPs ($\times 10^{10}$)
ResNet-50	2.352	8.610
VGG-16	1.472	32.141
Inception-v3	2.179	7.488
KSSNet	2.627	8.621
TDRG	4.873	8.609
SewerMl	0.916	3.001
Ours	2.469	8.610

FLOPs: FLoating-point OPerations

patches, which means that it can deal with about three original screen images per second. The inspection speed is acceptable in the real-world phone screen manufacturing industry. In the future, we will further optimize the code and model to meet the requirements of real-time inspection and high performance.

4.6 Network visualization

For qualitative analysis, we show the visualization for feature maps of different methods by using the class activation maps (CAMs) (Zhou et al., 2016). CAM is a method for interpreting the prediction decision made by CNN-based models. As illustrated in Fig. 7, the maps of our approach can highlight regions of low contrast or tiny-sized/incomplete defects and other illegible regions caused by the characteristic of ISIV. These regions are neglected by ResNet-50 (He et al., 2016). We give a few examples. Fig. 7a shows the original image to be visualized. Figs. 7b–7e are the visualized images of Fig. 7a. Figs. 7b and 7d both belong to the map of tiny-sized defect (i.e., class "pinhole"), but they correspond to ResNet-50 and our approach, respectively. We can see that the activate area of Fig. 7b is wider when compared with Fig. 7d, thereby reducing the contribution of the defective information area to the prediction score. The data results also show that our model can better identify the two tiny defects. To be specific, the score of ResNet-50 is 0.02 and that of our approach is 0.83 for the defect "pinhole" of this image. Fig. 7f shows the original image to be visualized. Figs. 7g–7j are the visualized images of Fig. 7f. Comparing Fig. 7h with Fig. 7j, the defect "scratch," which is of low contrast and incomplete, is ignored by ResNet-50. Fig. 7k shows the original image to be visualized. Figs. 7l–7o are the visualized images of Fig. 7k. As illustrated in Figs. 7m and 7o (i.e., the map of ResNet-50 and our approach), the defect "tin ash" is of too low contrast and different from others in the same class, resulting in positioning error of ResNet-50. In addition, it is clearly seen that the defective regions can be covered by the CAM masks of our approach. Therefore, our approach is quite helpful for capturing discriminative or meaningful regions by exploiting category-wise dependency, image-wise relations, and interactions between them.

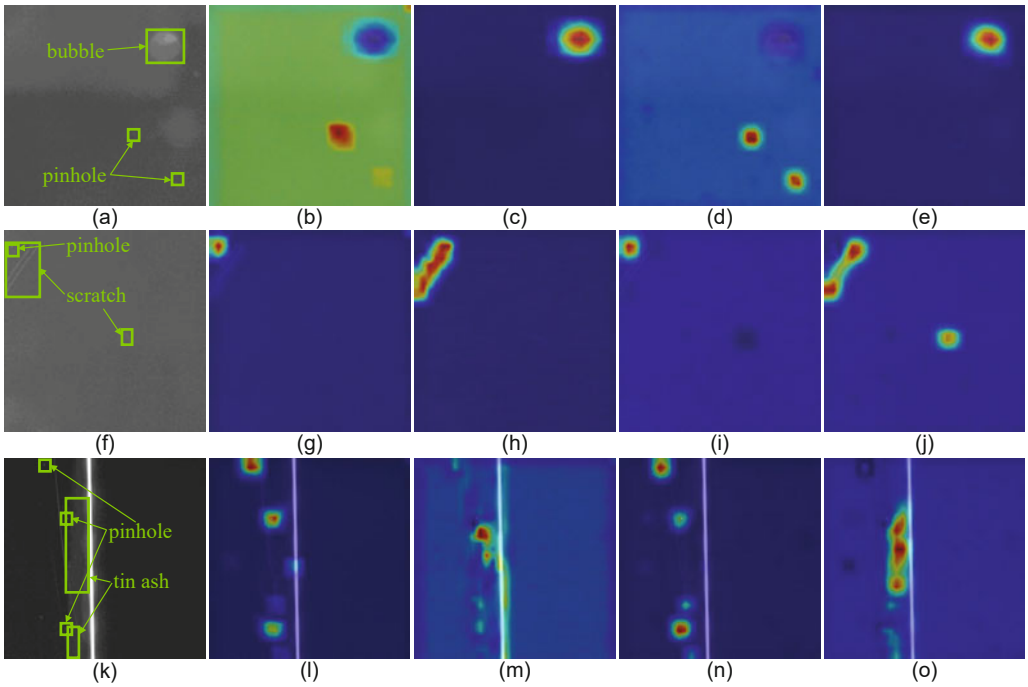

Fig. 7 Visualization for feature maps with a class activation map (CAM): defective images and manual annotation (a, f, k); visualization results of ResNet-50 for the defect "pinhole" (b, g, l); visualization results of ResNet-50 for the defects "bubble" (c), "scratch" (h), and "tin ash" (m); visualization results of ours for the defect "pinhole" (d, i, n); visualization results of ours for the defects "bubble" (e), "scratch" (j), and "tin ash" (o)

4.7 Generalization performance

To prove the generalization performance of our proposed model, we perform experiments on the steel dataset (https://www.kaggle.com/competitions/severstal-steel-defect-detection/data) created by Severstal. This dataset, with 12 568 images in total, contains four types of steel surface defects. We choose this dataset because it is also a multi-label defect dataset. Each image may be defect-free, or may have one or more defects. The dataset is divided in the proportion 8:1:1 for training, validation, and testing. The division is available on this website (https://github.com/CFZ1/The-division-for-kaggle-steel-dataset). All experimental results are shown in Table 9. Compared to other methods, our model achieves the best results on almost all metrics, which proves that it has good generalization capability.

Table 9 Comparisons with the state-of-the-art methods on the steel surface defect dataset

Method	Accuracy	Precision	Recall	F-measure
ResNet-50	**0.982**	0.915	0.877	0.896
VGG-16	0.974	0.850	0.848	0.849
Inception-v3	0.979	0.912	**0.928**	0.920
KSSNet	0.953	0.839	0.617	0.711
TDRG	0.978	0.896	0.853	0.874
SewerMl	0.954	0.813	0.707	0.756
Ours	**0.982**	**0.960**	0.885	**0.921**

The results in bold are the highest

5 Conclusions

Given the characteristics of mobile screen defect data from the real industrial pipelines and the difficulties in recognizing them, we proposed a two-stage defect inspection framework, consisting of the FC2-module and the GR-module. The former is able to classify defects, while the latter improves classification accuracy by reasoning semantics over image features with a category-to-category undirected graph. To solve the problem caused by ISIV of screen defects, we created a category-to-category undirected graph in the GR-module and used this graph to enhance image features. This category correlation in enhanced features increases the interclass distance and the intraclass correlation in the feature space. To obtain better node representations of the graph, we used contrastive learning and cosine similarity in the FC2-module. However, contrastive learning cannot be directly applied to the multi-label problem; so, we proposed the category pooling operator to solve it. Besides, the image-wise relations were exploited to improve low-quality image features. This helps the proposed network alleviate missed inspection caused by low contrast or tiny-size/incomplete defects. Moreover, we modeled the category dependencies by GCN and the co-occurrence patterns of defects. The experimental results on the mobile screen defect dataset showed that our model has better defect inspection performances: 97.7% accuracy and 97.3% F-measure. In addition, we visualized the feature maps of models with CAM, which showed that illegible defects can be detected by our approach.

Contributors

Chaofan ZHOU designed the research. Chaofan ZHOU and Meiqin LIU processed the data. Chaofan ZHOU drafted the paper. Senlin ZHANG helped organize the paper. Ping WEI and Badong CHEN revised and finalized the paper.

Compliance with ethics guidelines

Chaofan ZHOU, Meiqin LIU, Senlin ZHANG, Ping WEI, and Badong CHEN declare that they have no conflict of interest.

Data availability

Due to the nature of this research, participants of this study do not agree for mobile screen defect data to be shared publicly, so the supporting data are not available. The data used to validate the model generalization performance (i.e., the steel surface defect data) can be found on the web links given in this paper.

References

Bottou L, 2010. Large-scale machine learning with stochastic gradient descent. Proc 19th Int Conf on Computational Statistics, p.177-186. https://doi.org/10.1007/978-3-7908-2604-3_16

Chen T, Kornblith S, Norouzi M, et al., 2020. A simple framework for contrastive learning of visual representations. Proc 37th Int Conf on Machine Learning, p.1597-1607.

Chen ZM, Wei XS, Wang P, et al., 2019. Multi-label image recognition with graph convolutional networks. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.5177-5186. https://doi.org/10.1109/CVPR.2019.00532

Gidaris S, Komodakis N, 2018. Dynamic few-shot visual learning without forgetting. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.4367-4375. https://doi.org/10.1109/CVPR.2018.00459

Haurum JB, Moeslund TB, 2021. Sewer-ML: a multi-label sewer defect classification dataset and benchmark. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.13451-13462. https://doi.org/10.1109/CVPR46437.2021.01325

He KM, Zhang XY, Ren SQ, et al., 2016. Deep residual learning for image recognition. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.770-778. https://doi.org/10.1109/CVPR.2016.90

He KM, Fan HQ, Wu YX, et al., 2020. Momentum contrast for unsupervised visual representation learning. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.9726-9735. https://doi.org/10.1109/CVPR42600.2020.00975

Hjelm RD, Fedorov A, Lavoie-Marchildon S, et al., 2019. Learning deep representations by mutual information estimation and maximization. Proc 7th Int Conf on Learning Representations.

Khosla P, Teterwak P, Wang C, et al., 2020. Supervised contrastive learning. Proc 34th Conf on Neural Information Processing Systems, p.18661-18673.

Kong YH, Liu X, Zhao ZB, et al., 2022. Bolt defect classification algorithm based on knowledge graph and feature fusion. *Energy Rep*, 8(Suppl 1):856-863. https://doi.org/10.1016/j.egyr.2021.11.127

Lei J, Gao X, Feng ZL, et al., 2018. Scale insensitive and focus driven mobile screen defect detection in industry. *Neurocomputing*, 294:72-81. https://doi.org/10.1016/j.neucom.2018.03.013

Li CS, Zhang XM, Huang YJ, et al., 2020. A novel algorithm for defect extraction and classification of mobile phone screen based on machine vision. *Comput Ind Eng*, 146: 106530. https://doi.org/10.1016/j.cie.2020.106530

Lu Y, Ma L, Jiang HQ, 2020. A light CNN model for defect detection of LCD. In: Hung JC, Yen NY, Chang JW (Eds.), Frontier Computing. Springer, Singapore, p.10-19. https://doi.org/10.1007/978-981-15-3250-4_2

Park JY, Hwang Y, Lee D, et al., 2020. MarsNet: multi-label classification network for images of various sizes. *IEEE Access*, 8:21832-21846. https://doi.org/10.1109/ACCESS.2020.2969217

Paszke A, Gross S, Chintala S, et al., 2017. Automatic differentiation in PyTorch. Proc 31st Conf on Neural Information Processing Systems.

Simonyan K, Zisserman A, 2015. Very deep convolutional networks for large-scale image recognition. Proc 3rd Int Conf on Learning Representations.

Szegedy C, Vanhoucke V, Ioffe S, et al., 2016. Rethinking the inception architecture for computer vision. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.2818-2826. https://doi.org/10.1109/CVPR.2016.308

Wang T, Zhang C, Ding RW, et al., 2021. Mobile phone surface defect detection based on improved faster R-CNN. Proc 25th Int Conf on Pattern Recognition, p.9371-9377. https://doi.org/10.1109/ICPR48806.2021.9412119

Wang Y, He DL, Li F, et al., 2020. Multi-label classification with label graph superimposing. *Proc AAAI Conf Artif Intell*, 34(7):12265-12272. https://doi.org/10.1609/aaai.v34i07.6909

Wang YC, Gao L, Li XY, et al., 2021a. A new graph-based method for class imbalance in surface defect recognition. *IEEE Trans Instrum Meas*, 70:5007816. https://doi.org/10.1109/TIM.2021.3063755

Wang YC, Gao L, Gao YP, et al., 2021b. A new graph-based semi-supervised method for surface defect classification. *Rob Comput Integr Manuf*, 68:102083. https://doi.org/10.1016/j.rcim.2020.102083

Wei B, Hao KR, Gao L, et al., 2021. Bioinspired visual-integrated model for multilabel classification of textile defect images. *IEEE Trans Cognit Dev Syst*, 13(3):503-513. https://doi.org/10.1109/TCDS.2020.2977974

Wu ZR, Xiong YJ, Yu SX, et al., 2018. Unsupervised feature learning via non-parametric instance discrimination. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.3733-3742. https://doi.org/10.1109/CVPR.2018.00393

Xiao WW, Song KC, Liu J, et al., 2022. Graph embedding and optimal transport for few-shot classification of metal surface defect. *IEEE Trans Instrum Meas*, 71:5010310. https://doi.org/10.1109/TIM.2022.3169547

Xu H, Jiang CH, Liang XD, et al., 2019. Reasoning-RCNN: unifying adaptive global reasoning into large-scale object detection. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.6412-6421. https://doi.org/10.1109/CVPR.2019.00658

Yuan ZC, Zhang ZT, Su H, et al., 2018. Vision-based defect detection for mobile phone cover glass using deep neural networks. *Int J Precis Eng Manuf*, 19(6):801-810. https://doi.org/10.1007/s12541-018-0096-x

Zhang JB, Su H, Zou W, et al., 2021. CADN: a weakly supervised learning-based category-aware object detection network for surface defect detection. *Patt Recogn*, 109:107571. https://doi.org/10.1016/j.patcog.2020.107571

Zhao JW, Yan K, Zhao YF, et al., 2021. Transformer-based dual relation graph for multi-label image recognition. Proc IEEE/CVF Int Conf on Computer Vision, p.163-172. https://doi.org/10.1109/ICCV48922.2021.00023

Zhou BL, Khosla A, Lapedriza A, et al., 2016. Learning deep features for discriminative localization. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.2921-2929. https://doi.org/10.1109/CVPR.2016.319

Zhu Y, Ding RW, Huang WB, et al., 2022. HMFCA-Net: hierarchical multi-frequency based channel attention net for mobile phone surface defect detection. *Patt Recogn Lett*, 153:118-125. https://doi.org/10.1016/j.patrec.2021.11.029