





Research Article

When Transfer Learning Meets Dictionary Learning: A New Hybrid Method for Fast and Automatic Detection of Cracks on Concrete Surfaces

Si-Yi Chen ^{1,2}, You-Wu Wang ^{1,2}, Yi-Qing Ni ^{1,2} and Yang Zhang ^{1,2}

¹Department of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China

²National Rail Transit Electrification and Automation Engineering Technology Research Center (Hong Kong Branch), The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China

Correspondence should be addressed to You-Wu Wang; youwu.wang@polyu.edu.hk and Yi-Qing Ni; cseyqni@polyu.edu.hk

Received 11 March 2024; Revised 30 July 2024; Accepted 23 August 2024

Academic Editor: Wenai Shen

Copyright © 2024 Si-Yi Chen et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Cracks in civil structures are important signs of structural degradation and may indicate the inception of catastrophic failure. However, most of studies that have employed deep learning models for automatic crack detection are limited to high computational demand and require a large amount of labeled data. Long training time is not friendly to model update, and large amount of training data is usually unavailable in real applications. To bridge this gap, the innovation of this study lies in developing a hybrid method that comprises transfer learning (TL) and low-rank dictionary learning (LRDL) for fast crack detection on concrete surfaces. Benefiting from the availability of preextracted features in TL and a limited number of parameters in LRDL, the training time can be significantly minimized without GPU acceleration. Experimental results showed that the time for training a dictionary only takes 25.33 s. Moreover, this new hybrid method reduces the demand for labeled data during training. It achieved an accuracy of 99.68% with only 20% labeled data. Three large-scale images captured under varying conditions (e.g., uneven lighting conditions and very thin cracks) were further used to assess the crack detection performance. These advantages help to implement the proposed TL-LRDL method on resource-limited computers, such as battery-powered UAVs, UGVs, and scarce processing capability of AR headsets.

1. Introduction

During their long service lives, civil infrastructures experience various forms of degradation. Cracks often emerge as early indicators of structural deterioration. Their presence affects the appearance of the structure, reduces local stiffness, and jeopardizes integrity [1]. This highlights the necessity for the timely and accurate monitoring of cracks to ensure structural safety [2].

Traditionally, the manual inspection is widely applied for inspection of cracks on infrastructure surfaces, which relies on trained engineers with rich experience to record the irregular conditions on structures at regular intervals or after major disasters. However, owing to the size of the structure, manual visual inspection is time-consuming, labor-

intensive, and vulnerable to human errors. Furthermore, certain areas of the structure are inaccessible to inspection by humans owing to the challenges and potential risks involved. Therefore, to automate the inspection process and further minimize human costs, automated robotic inspection systems utilizing camera-equipped unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs) have been actively developed [3–5].

Robotic inspection systems are generally image-based. This means that to comprehend information from images, postprocessing combined with computer vision algorithms is typically required [6]. To date, various vision-based techniques have been proposed and developed for crack detection. Several surveys have also been conducted to review the existing image-based automatic crack detection

approaches from various perspectives [7–9]. Early image-based research for crack detection focused on image processing techniques (IPTs), including edge detection methods [10, 11], thresholding-based methods [12], percolation models [13], and graph theory-based methods [14, 15]. However, IPTs are susceptible to illumination conditions, distortion, and noise stemming from complex structural surfaces [16]. These conditions still limit the reliable detection of cracks using most of the existing IPTs.

Alternatively, other researchers have explored more adaptable methods for crack detection by integrating image feature extraction and machine learning (ML) techniques, and these methods have demonstrated robust noise resistance [17, 18]. In such studies, IPTs were utilized to extract crack-liked features. These extracted features were subsequently fed into ML classifier to distinguish between crack and noncrack images. Jahanshahi et al. [19, 20] specifically trained three classifiers using extracted morphological features to identify cracks of varying thicknesses. Zalama et al. [21] utilized visual features extracted from Gabor filters for road crack detection. They addressed the challenge of parameter selection by employing AdaBoost to combine a set of weak classifiers for feature extraction, resulting in improved performance. Chen et al. [22] adopted local binary patterns as a crack descriptor, in conjunction with SVM and Bayesian decision theory, to identify cracks on metallic surfaces. However, a crucial aspect of this process involved extracting robust hand-crafted features. Overextracted or false-extracted features often lead to numerous false positives in crack detection [23].

Deep convolutional neural networks (DCNNs) may overcome this problem, as their convolutional layers endow them with enhanced feature extraction ability [24, 25]. Unlike traditional hand-crafted feature extractors, DCNN convolutional kernels are learned automatically, granting them high flexibility and the capacity to extract more pertinent information relevant to the task at hand. Also, owing to their deep architectures, neural networks can efficiently capture global information by pooling knowledge from different layers and scales; this is challenging to accomplish using traditional hand-crafted feature extraction methods. For these reasons, DCNNs are emerging as powerful tools for detecting cracks in vision-based structural health monitoring (SHM).

As listed in Table 1, the DCNN-based crack detection methods can be divided into three categories: crack classification, crack localization, and pixel-level crack segmentation. For crack classification, captured images are divided into image blocks with the same size. These image blocks are then classified to determine whether they contain cracks or not. Therefore, it is indeed a binary classification task, where DCNNs are used to assign a binary label to each image patch. Cha et al. [26] developed a DCNN with four convolutional layers specially designed for concrete crack detection, incorporating the sliding window technique to analyze full-scale images. This approach achieved an accuracy of 97%, with the authors recommending the use of over 10,000 images for network training. Chen and Jahanshahi [27] proposed a DCNN with naïve Bayes data fusion scheme to

detect tiny cracks on metallic surfaces. The model was trained on a single GPU, and convergence was achieved after 70 epochs with 32, 535 s. To reduce the training time, Gopalakrishnan et al. [28, 29] used a pretrained VGG 16 network with transfer learning (TL) technique to detect cracks in pavement and UAV images. Among various ML classifiers (support vector machine, random forest, and logistic regression), it is reported that a single-layer neural network classifier with pretrained VGG 16 achieved the best classification results. Recently, Zhang and Yuen [30] developed an efficient crack detection framework that incorporates a feature-based broad learning system. Through the incorporation of incremental learning, this approach obviated the necessity for model retraining when updating the dataset. For real-world images, Pal et al. [47] highlighted the challenges associated with automatic crack detection in the presence of shadow effects. To mitigate their effects, a shadow augmentation technique is proposed to improve the classification accuracy [31]. Other classification networks for crack detection, including AlexNet, GoogleNet, ResNet, and SqueezeNet, can be found in these papers [32–34] and are summarized in Table 1.

Another type of crack detection method is based on object localization techniques, where the DCNNs are used to identify and locate cracks in images. Compared to the classification tasks with the fixed window size, crack localization possessed both object classification and localization capabilities, allowing for adaptive adjustment of the detection window size based on the identified object. Thus, it usually consists of a classification task and a regression task. Cha et al. [35] applied the faster region-based convolutional neural networks (faster R-CNN) to identify five types of surface damage, including concrete cracks, moderate and severe steel corrosion, bolt corrosion, and steel delamination. In their study, the training process based on the GPU device required was about 4 hours, whereas relying solely on a CPU extended the training time to around 4.5 days. Deng et al. [36] similarly used the faster R-CNN framework to identify cracks in real-world images featuring complex backgrounds. The results demonstrated that the method successfully differentiated cracks from handwriting script interferences on bridge surfaces. Jiang and Zhang [37] designed a wall-climbing UAV system used for crack inspection. For real-time crack detection, a tiny network integrated with Single Shot MultiBox Detector (SSD) and MobileNetV2 was trained and transplanted into a smartphone application. The proposed method facilitates crack inspection on a building and achieves a crack detection accuracy of 94.8%. You Only Look Once is a popular object detection algorithm [48], which has been developed from Version 1 (V1) to Version 10 (V10). Zou et al. [38] employed YOLOv4 network for postearthquake damage detection and safety evaluation. This approach successfully identified five common types of damage (fine cracks, wide cracks, concrete spalling, rebar exposure, and rebar bulking), showcasing high detection accuracy and potential for earthquake damage assessment. Qiu and Lau [39] utilized the YOLOv4-tiny, a more compact version of the YOLOv4 algorithm, for the detection of cracks in tiled sidewalks.

TABLE 1: Some DCNN-based crack detection methods.

	Reference	Backbone network	Training time (hours)
Crack classification	Cha et al. [26]	Shallow CNN	48 or 1.5 [#]
	Chen and Jahanshahi [27]	NB-CNN	9 [#]
	Gopalakrishnan et al. [28]	VGG 16	
	Gopalakrishnan et al. [29]	VGG 16	
	Zhang and Yuen [30]	ResNet50 -BLS	1.8 [#]
	Palevičius et al. [31]	AlexNet	
	Orinaitė [32]	AlexNet/SqueezeNet	5.9
	Ni et al. [33]	GoogleNet/ResNet 20	3 [#]
	Dung et al. [34]	VGG 16	
Crack localization	Cha et al. [35]	Faster R-CNN	108 or 4 [#]
	Deng et al. [36]	Faster R-CNN	57 [#]
	Jiang et al. [37]	SSDLite-MoblieNetV2	
	Zou et al. [38]	YOLOv4-D	
	Qiu et al. [39]	YOLOV4-tiny	
Pixel-level crack segmentation	Zhang et al. [40]	CrackNet	216 [#]
	Zhang et al. [41]	CrackNet-R	
	Dung and Anh [42]	FCN	
	Chen et al. [43]	SegNet	3 [#]
	Bhowmick et al. [44]	U-Net	
	Zhang et al. [45]	U-Net	1.1 [#]
	Lu et al. [46]	Swin transformer-U-Net	

Note. [#]Using GPU acceleration.

The above two types of crack detection methods (crack classification and localization) used the bounding boxes to locate and mark the detected cracks without further crack extraction. To directly detect cracks at the pixel level, some DCNN-based crack segmentation methods have been proposed. In such an end-to-end framework, detailed segmentation crack maps can be automated and generated from input images. Zhang et al. [40] proposed a CrackNet to detect cracks on asphalt pavement. Unlike conventional CNNs, CrackNet does not include any pooling layers; thus, all pixels in images can be predicted. Later, a modified version called CrackNet-R was proposed for fully automated pixel-level crack detection [41]. Dung and Anh [42] developed a crack detection method based on a fully convolutional network (FCN) for semantic segmentation of concrete crack images. They selected different DCNNs as encoders to extract features and demonstrated that VGG16 performed better than InceptionV3 and ResNet for feature extraction. Similarly, based on an encoder-decoder structure, a novel U-Net architecture was recently established to improve the accuracy, effectiveness, and robustness in crack segmentation [44–46, 49, 50]. Bhowmick et al. [44] used a U-Net architecture for pixel-level crack segmentation and validated their approach by processing video data captured from a UAV-based camera. Zhang et al. [45] proposed an improved U-Net-based model to identify crack, where a new loss function was applied to solve the class imbalance problem. Recently, with the transformer and self-attention mechanism shown promising performance in computer vision, Lu et al. [46] integrated the Swin Transformer module into U-Net for crack segmentation. In this method, the Swin Transformer serves as the encoder segment to extract features.

Although DCNNs have shown promising advancements than traditional methods, there are several challenges associated with their use. First, it usually requires large amounts of annotated data for classification, identification, and segmentation tasks. Especially for the segmentation task, which requires manual annotation of every pixel in the image, the labeling process can be costly. Second, even when a well-annotated dataset is available, training a network from scratch can take hours or even days, with the use of high-performance computing systems, as presented in Table 1. Third, as the monitoring data continue to grow, retraining the DCNN model becomes necessary using the updated dataset. Long training time is not friendly to model update with the new updated dataset. These limitations affect the practical applicability of the existing methods.

In this study, a hybrid method is developed for crack detection on concrete surfaces, intended to overcome above limitations. The proposed method comprises transfer learning (TL) and low-rank dictionary learning (LRDL), which is hereinafter referred to as TL-LRDL. The main contributions of this study are summarized as follows: (1) The present study represents the first attempt to combine TL and LRDL in SHM for fast crack detection on concrete surfaces. (2) Benefiting from the availability of preextracted features in TL and only a limited number of parameters in LRDL, the training time can be significantly minimized without requiring GPU acceleration. (3) TL-LRDL enables the reuse of the established DCNN model without necessitating additional network training, thereby reducing the demand for labeled data during training. Consequently, these advantages help to implement the proposed TL-LRDL method on edge or resource-limited devices, such as battery-powered UAVs, UGVs, and scarce processing capability of

AR headsets. Furthermore, the fast-training process provides a potential solution to complete quick model updates when the dataset is updated.

In the remainder of this article, Section 2 introduces the proposed crack detection method combining TL and LRDL. Section 3 presents the performance of the trained classification model and the implementation results on large-scale images. Section 4 compares the detection performance of the proposed method with those of existing DCNN methods and two commonly used edge detection methods, followed by the conclusion in Section 5.

2. Proposed Methodology

By incorporating TL and LRDL, this research introduces a novel method for detecting cracks, referred to TL-LRDL. The proposed TL-LRDL is fast, autonomous, and easy to implement. It involves reusing a well-trained DCNN model and “transferring” its learning ability to a crack detection task. According to the transferred features, a new classifier for crack detection via LRDL is generated. The proposed TL-LRDL method is described below in detail.

2.1. Pretrained ResNet18 Model. Over the past decade, various DCNN architectures have emerged with increasing depth and complexity. Given that crack detection constitutes a relatively simpler binary classification task (crack or noncrack), a simple yet effective architecture (ResNet18) is chosen for this study. ResNet18 (i.e., residual network with 18 layers) was introduced in 2015; the model has secured the top spot in image classification competitions (ILSVRC 2015 and ILSVRC and COCO 2015). Different from other DCNNs equipped with stacked convolutional layers (such as VGG or AlexNet), ResNet’s most notable feature is its incorporation of additional residual (or skip) connections. These connections play a crucial role in addressing degradation in deep networks [51].

The ResNet18 model comprises two core components: the model architecture and the trainable parameters (weights) (Figure 1). The model architecture includes 17 convolutional layers along with a fully connected (FC) layer. The former layers utilize convolution kernels of sizes 7×7 or 3×3 to extract features from input images. By expanding depth through the addition of more convolutional layers, consistent with the concept of a “very deep network” [25], ResNet18 can effectively extract features from images. The final FC layer establishes the relationship between the extracted image features and outcomes of the classification.

The model includes 11.6 million trainable parameters. These parameters were determined through the original training of the ResNet18 architecture on the ImageNet dataset, which comprises over 3.2 million natural images distributed across 5,247 categories. Consequently, the pretrained ResNet18 model on ImageNet exhibits a recognition capability encompassing 1,000 categories (source task). Since the pretrained ResNet18 model has acquired the capability to extract distinguishing features that can

differentiate one image class from another, they prove to have very good generality on another dataset, which in turn creates an environment conducive to TL.

2.2. Feature Transfer. In SHM applications, obtaining a large-scale dataset such as ImageNet to train an entire DCNN (ResNet18) from scratch is impractical, and the complete training process is time-consuming. TL has been proposed as a powerful tool to avoid this challenge. Its aim is to leverage knowledge from the source domain to improve the training efficiency or minimize the number of labeled data required in the target domain.

TL typically involves two common strategies: feature extraction and fine-tuning [52]. In the feature extraction approach, all parameters and weights preceding the FC layers are fixed. These fixed and unchanged parameters are represented as learned knowledge from the source task and then transferred from the source domain to the target domain. The features extracted by keeping these parameters are used to train a classifier for predicting labels in the target domain. In contrast, the fine-tuning strategy involves retraining certain parts of the DCNN. In this strategy, only some parameters from the initial layers are fixed, while higher convolutional blocks, together with the FC layers, are retrained through gradient descent and backpropagation. Depending on factors such as data size and problem complexity, either of these two strategies can be applied in different scenarios.

In this study, the feature extraction strategy for TL is adopted for two reasons: (1) The original training dataset for the pretrained ResNet18 model is ImageNet, encompassing a multitude of images associated with the civil engineering field. For crack detection task (the target task) in structural engineering domain, the source and target domains are related. Thus, such a pretrained ResNet18 model on ImageNet might have learned features that can distinguish crack images from noncrack images. (2) Feature extraction is faster than fine-tuning, as the former does not require additional network training.

Specifically, as depicted in Figure 2, by maintaining the parameters prior to the FC layer as fixed, the pretrained ResNet18 model serves as a feature extractor in this study. The outputs from the final layer preceding the FC layer are extracted as semantic features, which is subsequently transferred to the target domain. Each image yields semantic features with a final dimension of 512. These semantic features are then utilized to train a newly generated classifier via LRDL. The LRDL phase benefits from the availability of preextracted features and a limited number of parameters, which considerably reduces the training time and the required number of labeled images. Thus, the overall procedure becomes a lightweight framework designed for execution on computationally limited platforms.

2.3. Low-Rank Dictionary Learning (LRDL) for Crack Detection and Classification. Using the transferred features, we train a classifier to conduct the crack detection task (target task) via LRDL.

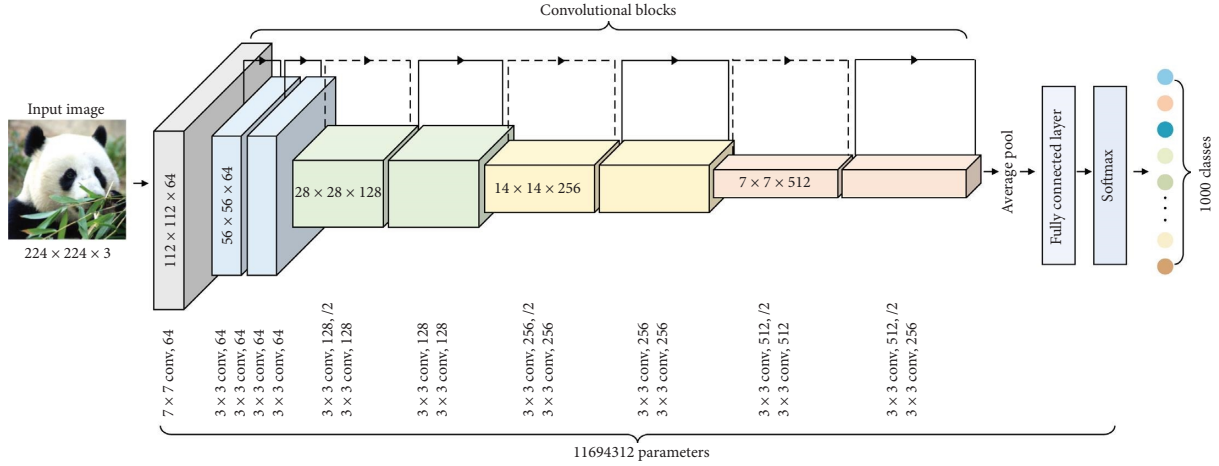


FIGURE 1: Pretrained ResNet18 model.

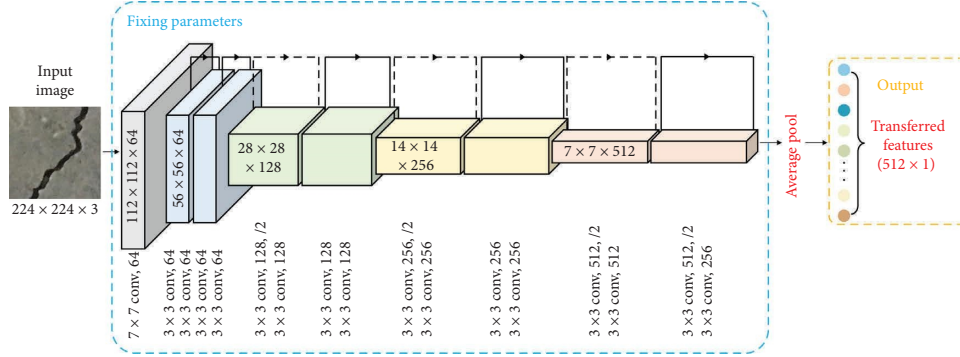


FIGURE 2: ResNet18 with TL.

2.3.1. *Low-Rank Representation (LRR)*. LRR, similar to sparse representation classification (SRC) model, belongs to representation-based classification methods. Due to the promising classification performance, they have been widely used for facial recognition and image classification. However, the application of LRR in SHM for the detection of cracks remains relatively unexplored.

Different from SRC model with ℓ_1 norm, LRR with nuclear norm is more robust to noise and outliers, as the low-rank constraint is superior at capturing the global structure of the data. The basic idea of LRR is to represent the query data as a linear combination of several atoms from an overcomplete dictionary encompassing all available classes. As the representation coefficients can well reflect the similarities among data, the resulting representation matrix contains critical class information, making it useful for classification purposes. A more comprehensive mathematical explanation is provided subsequently [53].

For a given testing set \mathbf{Y} , $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N] \in \mathbb{R}^{d \times N}$ is defined as a set of N test images in a d -dimensional feature space. Each column $\mathbf{y}_i \in \mathbb{R}^d$ denotes the transferred features of image i , which is the outputs of the pretrained ResNet18 model before the FC layer. As mentioned earlier, the final feature dimension of each image is 512 (i.e., $d = 512$). With

an overcomplete dictionary \mathbf{D} , each test image in feature space $\mathbf{y}_i \in \mathbb{R}^d$ can be expressed as a linear combination of the atoms in the dictionary. As such, \mathbf{Y} can be expressed as

$$\mathbf{Y} = \mathbf{D}\mathbf{Z} + \mathbf{E}, \quad (1)$$

where $\mathbf{D} = [\mathbf{D}_1, \mathbf{D}_2, \dots, \mathbf{D}_K] \in \mathbb{R}^{d \times m}$ denotes a dictionary from all K classes, and \mathbf{D}_i is a subdictionary corresponding to the class i . m denotes the total number of dictionary atoms. $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N]$ denotes the representation matrix, where each column \mathbf{z}_i corresponds to the representation coefficients of \mathbf{y}_i , and \mathbf{E} stands for the sparse noise component.

If a test image $\mathbf{y}_i \in \mathbb{R}^d$ belongs to the class k , then ideally, it can be well represented by its own subdictionary \mathbf{D}_k , and the representation coefficients for \mathbf{D}_j ($j \neq k$) are nearly all zero. Thus, the positions of nonzero elements in \mathbf{z}_i reveal the class information of \mathbf{y}_i , and the optimal representation matrix \mathbf{Z} for \mathbf{Y} should be block-diagonal as follows [54]:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Z}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Z}_K \end{bmatrix}. \quad (2)$$

To achieve the above block-diagonal structure in \mathbf{Z} , the low-rank constraint is imposed on \mathbf{Z} . Compared with the sparse constraint, the low-rank constraint is superior at capturing the global structure of the data because it exploits correlations among its columns [55]. Then, equation (1) can be solved using the following LRR optimization problem:

$$\begin{aligned} \min_{\mathbf{Z}, \mathbf{E}} \quad & \text{rank}(\mathbf{Z}) + \lambda \|\mathbf{E}\|_0 \\ \text{s.t.} \quad & \mathbf{Y} = \mathbf{D}\mathbf{Z} + \mathbf{E}, \end{aligned} \quad (3)$$

where $\lambda > 0$ is a trade-off parameter that balances the contribution of the rank term and the noise component. Owing to the discrete nature of the rank operation and the ℓ_0 norm, equation (3) is a nonconvex problem that is difficult to solve. Therefore, the convex relaxation of equation (3) is expressed as follows:

$$\begin{aligned} \min_{\mathbf{Z}, \mathbf{E}} \quad & \|\mathbf{Z}\|_* + \lambda \|\mathbf{E}\|_1 \\ \text{s.t.} \quad & \mathbf{Y} = \mathbf{D}\mathbf{Z} + \mathbf{E}, \end{aligned} \quad (4)$$

where $\|\cdot\|_*$ denotes the nuclear norm of a matrix, serving as an effective surrogate for the rank minimization problem, and $\|\cdot\|_1$ denotes the ℓ_1 norm as the convex relaxation of the ℓ_0 norm.

2.3.2. Dictionary Learning (DL). Dictionary quality holds significant importance for classification within the framework of LRR, as it profoundly influences the discriminative nature of the resulting representation matrix \mathbf{Z} . Generally, most classification methods based on LRR utilize the entire training set as the dictionary to learn the LRR. Nevertheless, a dictionary composed of all training data features excessive redundant atoms, which increases the computational cost and reduces the discriminative capacity for classification. Moreover, the dictionary \mathbf{D} is fixed in the solving process, as shown in equation (4). If the dictionary contains noise or outliers, the test images $\mathbf{y}_i \in \mathbb{R}^d$ cannot be represented well by a polluted dictionary. Therefore, it is necessary to integrate a DL progress into equation (4), instead of fixing dictionary atoms.

Typically, DL approaches can be divided into two main categories [56, 57]: (1) the acquisition of a dictionary founded on mathematical models, such as wavelet, contourlet, bandelet, and wavelet packets, and (2) the construction of a dictionary that behave well in the training dataset. In this study, the latter way is selected owing to its ability to yield excellent classification performance in most practical scenarios. The mathematical model for DL can be described as follows:

$$\begin{aligned} \min_{\mathbf{Z}, \mathbf{D}, \mathbf{E}} \quad & \|\mathbf{Z}\|_* + \lambda \|\mathbf{E}\|_1 + \frac{\gamma}{2} \|\mathbf{D}\|_F^2 \\ \text{s.t.} \quad & \mathbf{X} = \mathbf{D}\mathbf{Z} + \mathbf{E}, \end{aligned} \quad (5)$$

where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ denotes the training set consisting of n training images, and each column \mathbf{x}_i represents the features of a training image transferred from the

pretrained ResNet18 model; $(\gamma/2)\|\mathbf{D}\|_F^2$ is included to prevent scale change during the DL process. The model for LRR with DL (called LRDL) is shown in Figure 3.

2.3.3. Locality Constraint for LRDL. Despite LRR's robust capability to explore the global structure of a dataset, it overlooks the local manifold structure existing between pairs of images. This local structure is also valuable for classification purposes [58]. To preserve this local structure, we employ the Euclidean distance to gauge local similarity:

$$W_{i,j} = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2, \quad (6)$$

where $W_{i,j}$ is the distance between \mathbf{x}_i and \mathbf{x}_j . It measures the similarity between two images; that is, the smaller the $W_{i,j}$ is, the more similar the two images are, while the larger the $W_{i,j}$ is, the more different the two images are. Thus, considering the locality relationship can result in better classification performance. The locality constraint, serving as a penalty term for LRDL, is introduced as follows:

$$\|\mathbf{W} \odot \mathbf{Z}\|_1, \quad (7)$$

where \odot denotes the Hadamard product, which means elementwise multiplication. If \mathbf{W} is an all-one matrix, $\|\mathbf{W} \odot \mathbf{Z}\|_1$ reduces to $\|\mathbf{Z}\|_1$, which becomes traditional and "unweighted" ℓ_1 norm on \mathbf{Z} . The ℓ_1 norm is a convex surrogate for the ℓ_0 norm, commonly used to measure sparsity in the solution. The difference between $\|\mathbf{W} \odot \mathbf{Z}\|_1$ and $\|\mathbf{Z}\|_1$ lies in an additional weight matrix \mathbf{W} that is added. As defined in equation (6), $W_{i,j}$ is the Euclidean distance between \mathbf{x}_i and \mathbf{x}_j , measuring the similarity between two images. All $W_{i,j}$ in \mathbf{W} are positive values, and these weights $W_{i,j}$ are inversely proportional to the $Z_{i,j}$. Thus, by Hadamard product between \mathbf{W} and \mathbf{Z} , this whole term $\|\mathbf{W} \odot \mathbf{Z}\|_1$ can be implicitly considered as a weighted ℓ_1 norm. And the large weights $W_{i,j}$ could be used to discourage nonzero entries in \mathbf{Z} , while small weights could be used to encourage nonzero entries. In this way, $\|\mathbf{W} \odot \mathbf{Z}\|_1$ captures both the sparsity of \mathbf{Z} and the similarity between two images. Finally, this locality constraint term $\|\mathbf{W} \odot \mathbf{Z}\|_1$ is integrated into equation (5) to enable the learning of a discriminative dictionary and representation as follows:

$$\begin{aligned} \min_{\mathbf{Z}, \mathbf{D}, \mathbf{E}} \quad & \|\mathbf{Z}\|_* + \lambda \|\mathbf{E}\|_1 + \alpha \|\mathbf{W} \odot \mathbf{Z}\|_1 + \frac{\gamma}{2} \|\mathbf{D}\|_F^2 \\ \text{s.t.} \quad & \mathbf{X} = \mathbf{D}\mathbf{Z} + \mathbf{E}, \end{aligned} \quad (8)$$

where α controls the contribution of the locality constraint term. The optimization algorithm to solve equation (8) is described in Appendix A.

By integrating a locality constraint in the training process, a compact and discriminative dictionary \mathbf{D} can be learned from all the training data using equation (8). This learned dictionary promotes similar representations for images within the same class and distinct representations for those from different classes, thereby enhancing robustness of the classification performance in detecting cracks.

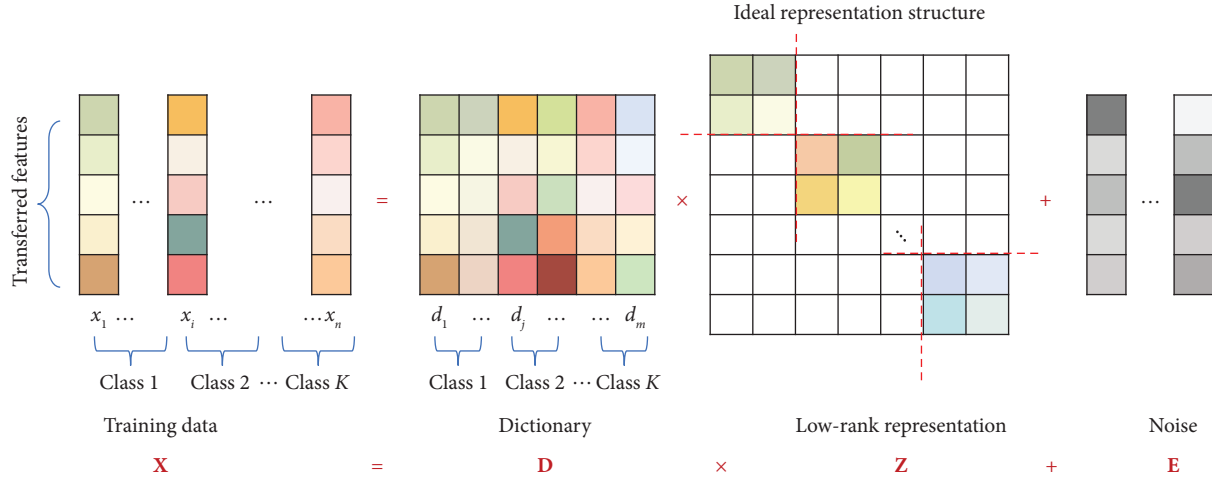


FIGURE 3: Illustration of the LRDL model. The colored and white blocks indicate the nonzero and zero values, respectively, and the grey blocks indicate the values close to zero.

2.3.4. Classification Using Multivariate Ridge Regression Model. After a discriminative dictionary \mathbf{D} is learned from equation (8), the low-rank representation matrix of test images $\mathbf{Z}_{\text{Test}} = [\mathbf{z}_1^{\text{Test}}, \mathbf{z}_2^{\text{Test}}, \dots, \mathbf{z}_N^{\text{Test}}] \in \mathbb{R}^{m \times N}$ can be obtained by solving equation (4). Each column $\mathbf{z}_i^{\text{Test}}$ in the matrix denotes the representation coefficients of test image i , which contains critical class information and could be used for further classification.

Here, the multivariate ridge regression model [59] is employed to derive a basic linear classifier. According to the representation matrix $\mathbf{Z}_{\text{Train}}$ of training images \mathbf{X} from equation (8), the linear classifier \mathbf{L} is obtained as follows:

$$\mathbf{L} = \operatorname{argmin}_{\mathbf{L}} \|\mathbf{H} - \mathbf{L}\mathbf{Z}_{\text{Train}}\| + \eta \|\mathbf{L}\|_F^2, \quad (9)$$

where \mathbf{H} is a matrix representing the class labels of training data \mathbf{X} and η controls the weight of the regularization term. Equation (9) is a convex problem, and its closed-form solution is expressed as follows:

$$\mathbf{L} = \mathbf{H}\mathbf{Z}_{\text{Train}}^T (\mathbf{Z}_{\text{Train}}\mathbf{Z}_{\text{Train}}^T + \eta\mathbf{I})^{-1}. \quad (10)$$

Then, the class label for a test image i is determined by

$$k^* = \operatorname{argmax} \mathbf{L}\mathbf{z}_i^{\text{Test}}, \quad (11)$$

where k^* is the predicted label corresponding to the classifier with the largest output.

In summary, the proposed TL-LRDL method can be divided into four steps for crack detection and classification:

- (1) Utilizing the pretrained ResNet18 model to extract and transfer the “learned” features to the target domain
- (2) Using the transferred features to learn a compact and discriminative dictionary \mathbf{D} from training dataset \mathbf{X} by solving equation (8) (see Appendix A)
- (3) Using the learned dictionary \mathbf{D} to obtain the low-rank representation matrix for test images \mathbf{Z}_{Test} according to equation (4)

- (4) Conducting crack detection and classification on these discriminative representations \mathbf{Z}_{Test} , with class labels determined by equations (10) and (11)

The overall framework of the proposed crack detection method is illustrated in Figure 4.

3. Results

3.1. Dataset. Images of concrete surfaces with cracks are obtained from a publicly available and realistic crack dataset [60]. The dataset comprises 40,000 RGB image patches, each with a resolution of 227×227 pixels. These image patches were originally extracted from 458 full-scale images (4032×3024 pixels) captured at the Middle East Technical University Campus. The 40,000 images are evenly distributed across two distinct classes, “crack” and “noncrack,” for the classification task. Both classes contain 20,000 images each. The full images have a high surface finish and illumination condition variance, and no data augmentation, including image rotation or flipping, was applied to the dataset [42]. Figure 5 shows some typical data samples. To verify the efficacy of the proposed method under a few labeled data, we adopt a training-to-testing set ratio of 20%:80% (8,000 images for training and 32,000 for testing). The training and testing sets are class-balanced and contain an equal number of cracked and noncracked images. For further details, Table 2 provides a summary of the dataset used.

ResNet18 has a fixed input image size of 224×224 pixels, which differs from the image size in the dataset (227×227 pixels). Thus, before the images are fed into the above network, they are first rescaled to the corresponding size to be compatible with the ResNet18 architecture.

ResNet18 requires an input image size of 224×224 pixels, which is not the same as the dataset’s image size of 227 by 227 pixels. Therefore, to ensure compatibility with the ResNet18 structure, the images from the dataset must be resized to the appropriate dimensions prior to their utilization in the network for feature extraction and transfer learning.

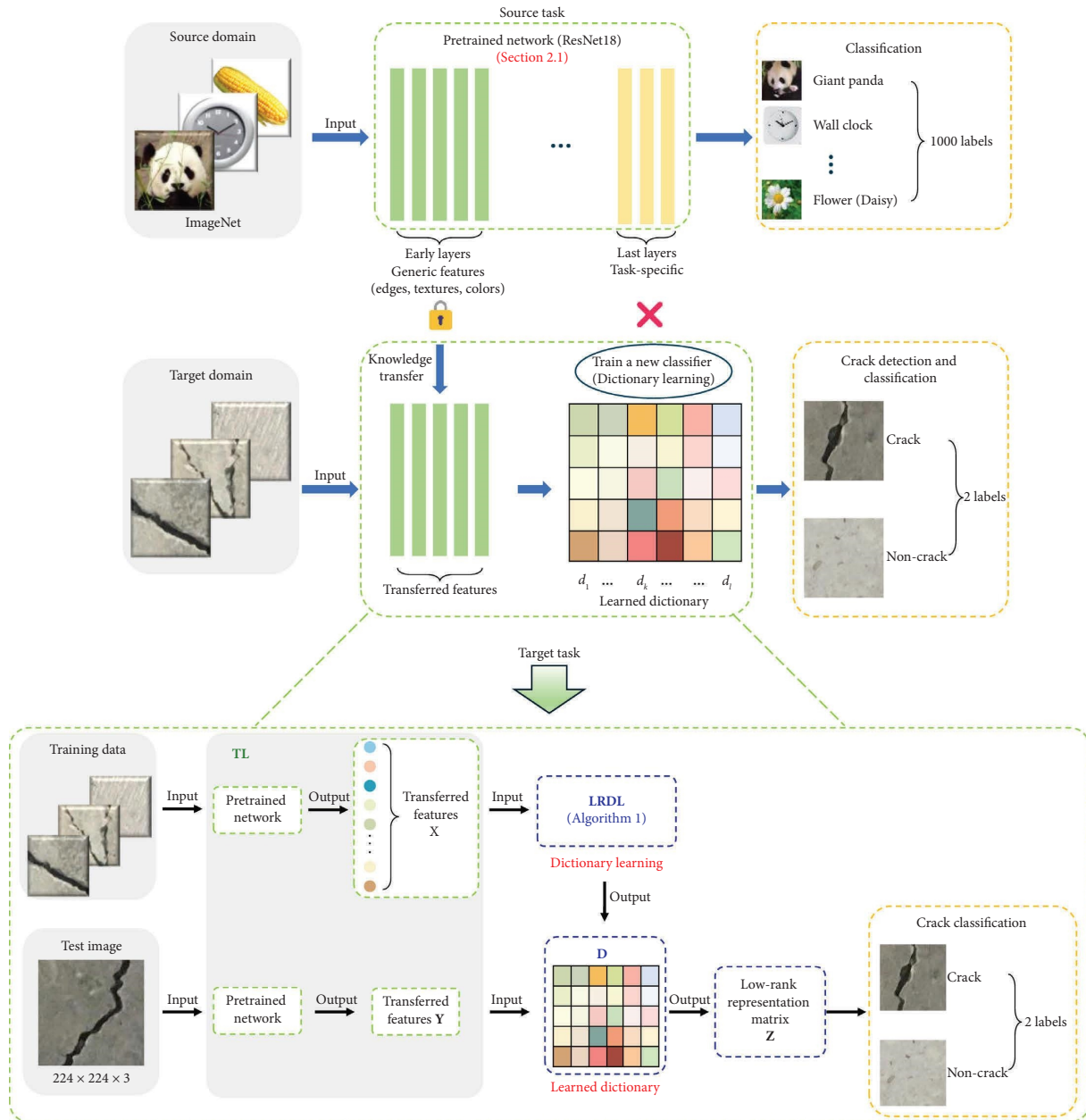


FIGURE 4: Overall framework of crack detection via TL-LRDL (the yellow lock indicates the transfer of generic features from the pretrained network; the red cross indicates no knowledge transfer).

The present study is conducted on MATLAB 2022b on a computer equipped with an Intel® Core™ i7-8700 @3.2 GHz CPU and 32 GB DDR4 installed memory (RAM), with no discrete graphic processing unit (GPU). The proposed crack detection method is implemented using the “Deep Learning Toolbox™,” “Computer Vision Toolbox™,” and “Image Processing Toolbox™” within MATLAB.

3.2. Performance Analysis of Trained Classification Model. Following the application of the proposed TL-LRDL to the training dataset, a classification model for crack detection is generated. Different from most deep learning methods that require hours or days for model training, the proposed TL-LRDL has far shortened the training time. As

a main merit of the proposed method, TL-LRDL can realize efficient training without GPU acceleration. In this study, the entire training process only took about 163.33 s (less than 3 min), where the time for ResNet18 feature extraction was about 137 s and the time for training a dictionary only took about 25.33 s. This advantage provides a potential solution to complete quick model updates with the updated datasets.

Then, the performance of the trained classifier is evaluated using the testing set. Through comparison of the predicted and ground-truth images, the correct and incorrect classifications are obtained to form the confusion matrix (Figure 6). The proposed TL-LRDL achieves almost perfect classification results, with only 11 false positives

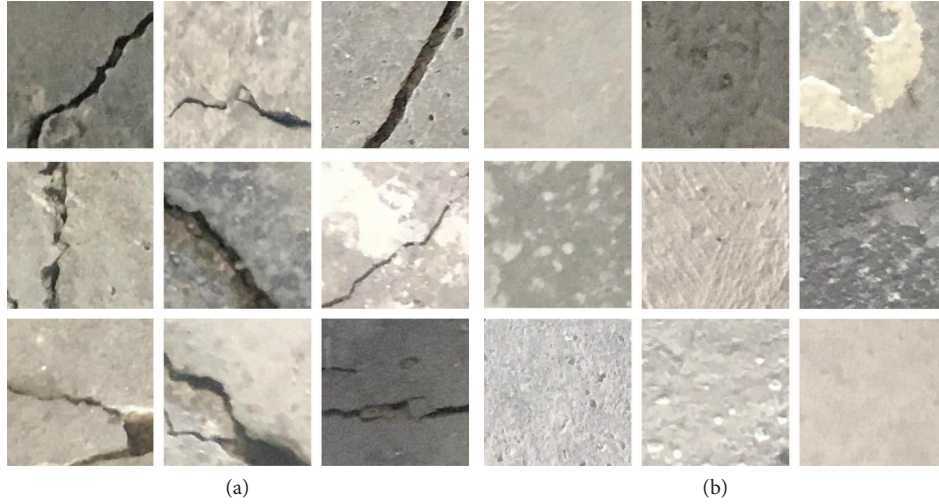


FIGURE 5: Portion of image samples in the dataset: (a) with cracks; (b) without cracks (size: 227×227 pixels).

TABLE 2: Summary of concrete crack image dataset.

Material	Number of images	Image size (pixels)	Crack	Noncrack	Training set	Testing set
Concrete	40,000	227×227	20,000	20,000	8,000	32,000

Predicted Class	Crack	TP 15907 (99.9%)	FN 93 (0.6%)
	Non-crack	FP 11 (0.1%)	TN 15989 (99.4%)
		Crack	Non-crack
		True Class	

FIGURE 6: Confusion matrix.

(FPs) and 93 false negatives (FNs) detected among the total of 32,000 test images in the testing set.

The accuracy of the trained classification model can be further quantified using several evaluation metrics:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \quad (12)$$

$$\text{precision} = \frac{TP}{TP + FP}, \quad (13)$$

$$\text{recall} = \frac{TP}{TP + FN}, \quad (14)$$

$$F1 - \text{score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \quad (15)$$

where TP, TN, FP, and FN represent the counts of test images corresponding to true positives, true negatives, false positives, and false negatives, respectively.

Referring to the confusion matrix in Figure 6, the obtained metrics are listed in Table 3. The classification accuracies for both the training and testing sets are exceptionally high, reaching 99.71% and 99.68%, respectively. In addition, the precision, recall, and F1-score values are 99.93%, 99.42%, and 99.67%, respectively. These

TABLE 3: Classification performance of trained classification model.

Training accuracy (%)	Testing accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Training time (LRDL) (s)
99.71	99.68	99.93	99.42	99.67	25.33

results demonstrated that (1) this new hybrid method reduces the demand for labeled data during training. TL-LRDL trained on 20% labeled data could achieve a high accuracy of 99.68%; (2) the time for training the crack detection model is extremely short with high efficiency. As mentioned, it will benefit for model update.

3.3. Implementation on Large-Scale Images. The generalization ability of the trained classification model is demonstrated using three newly collected large-scale images not used in the training process. These images are taken under various conditions by a hand-level device (IPad10) from the walls of a concrete building at the Hong Kong Polytechnic University. Each image has a resolution of 4032×3024 pixels.

Figure 7 illustrates the framework of the proposed method implemented on large-scale images. As shown in this figure, the developed classification model is first applied to process the test images by implementing a sliding window across the entire image. Regions within these local windows that the model identifies as containing cracks are then highlighted in the output image using bounding boxes. Then, based on the identified windows with cracks, a detailed analysis is conducted to derive geometric measurements of the cracks, such as length, width, and area. The detailed detection and measurement process are presented below.

3.3.1. Crack Detection and Localization. Given that most images in practice have sizes larger than 224×224 pixels, to detect cracks on large-scale images with 4032×3024 pixels, sliding window technique is used here. The window size is set to 224×216 pixels. Each large-scale image needs to be scanned only once, with no overlapping areas. The miniature image confined within the sliding window is input into the classification model. When the model predicts that it is a “crack,” this small image patch is marked with a bounding box. For each large-scale image of size 4032×3024 , ResNet requires about 3 s to extract features, and the time for output generation is only approximately 0.88–0.92 s. Hence, the inference time for each small image patch is about 0.015 s on average.

Figure 8 illustrates the crack detection outcomes for the three tested large-scale images. The proposed method accurately identifies and locates the cracks within the images. An image containing pronounced cracks with uneven lighting conditions is tested (Figure 8(a)). The successful detection of cracks in Figure 8(a) demonstrates that the proposed method is robust to lighting conditions. To study the effects of crack widths, one image containing thin cracks and another with hairline cracks are tested (Figures 8(b) and 8(c)). The proposed method identifies and localizes the very thin cracks, and even some hairline cracks are detected. Moreover, some cracks on the edges of sliding windows are

successfully detected by the proposed method (Figures 8(a), 8(b), and 8(c)). Compared to the images with cracks in the middle of sliding windows, cracks on the edges are more difficult to be identified.

The detected crack image patches are subsequently compared with the ground-truth labels, and their results are color-coded (Figure 9). Red highlights denote incorrect predictions, where a noncrack image is mistakenly classified as a crack (FP), while missed crack instances (FN) are indicated in green. In Figure 9(a), four sliding windows are classified as FP and FN regions. In Figure 9(c), two FN regions exist, and six sliding windows are classified as FP regions. Compared to Figures 9(a) and 9(c), the results in Figure 9(b) are more accurate, with only one sliding window being incorrectly identified as crack. As shown in Table 4, the detection accuracies achieved on the three large-scale images are 96.83%, 99.60%, and 96.83%. These findings showcase the capability of the proposed method to accurately detect cracks on structural surfaces, making the model a viable alternative to manual inspection.

3.3.2. Crack Pixel-Level Segmentation and Geometric Measurement. The combination of the proposed classification model and the sliding window technique enables the satisfactory detection of cracks within images. However, the necessary details required for quantifying damage, such as crack width and length, cannot be inferred from the obtained bounding boxes. Thus, image segmentation is needed for pixel-level crack segmentation.

In this paper, image processing techniques, such as median filtering and Otsu’s threshold method [61], are used to segment cracks at the pixel level within windows. Specifically, the developed classification model initially identifies windows containing cracks (Figure 8). Subsequently, by discarding the windows not containing cracks, the segmentation results of selected window are displayed on a white mask of the same size as the original image according to their coordinate information, as shown in Figure 10 (left column). Hence, the binary crack map is actually the intermediate results, generated after crack detection (Figure 8) and further used for crack geometric measurements (Figure 10: right column).

The segmented crack images are then undergo further processing to obtain geometric measurements of cracks, such as its width, length, and area (Table 5). Morphological operations from image processing are employed to assess the geometric properties of the cracks [20, 44]. The crack area is calculated by tallying the number of pixels within the crack map. The crack length is derived through the skeletonization process applied to the crack map, utilizing single-pixel crack skeletons that delineate the centerline of cracks. Additionally, the Euclidean distance transform of the crack map generates

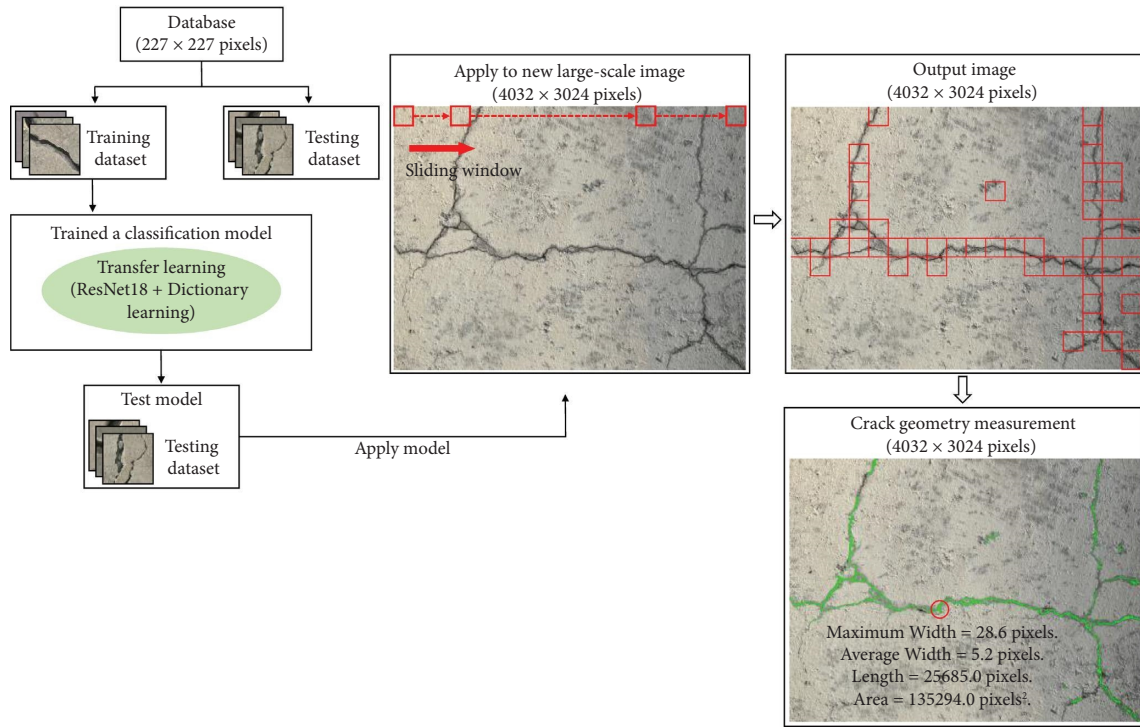


FIGURE 7: Framework of the proposed method implemented on large-scale images.

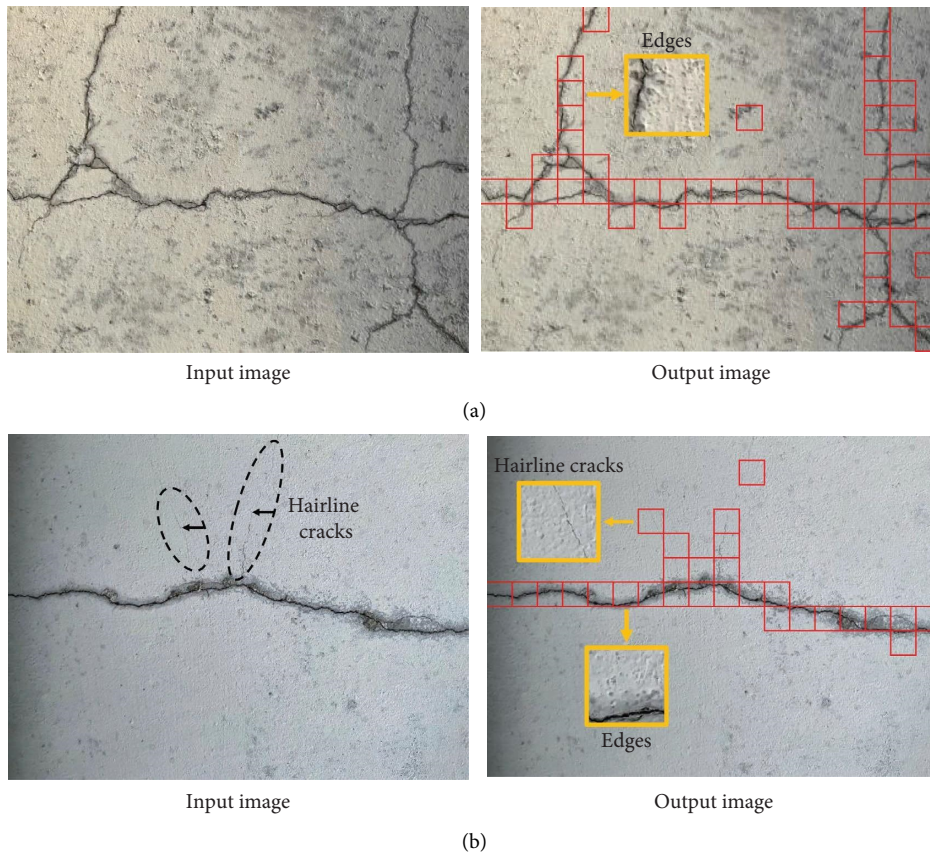
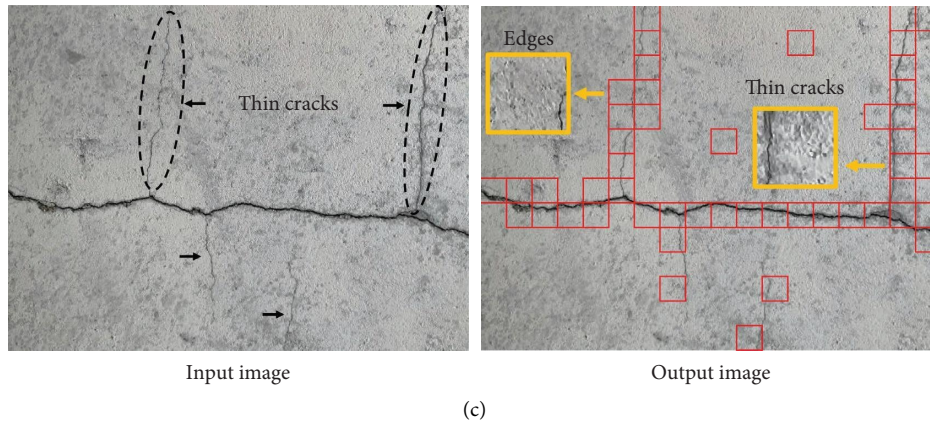


FIGURE 8: Continued.

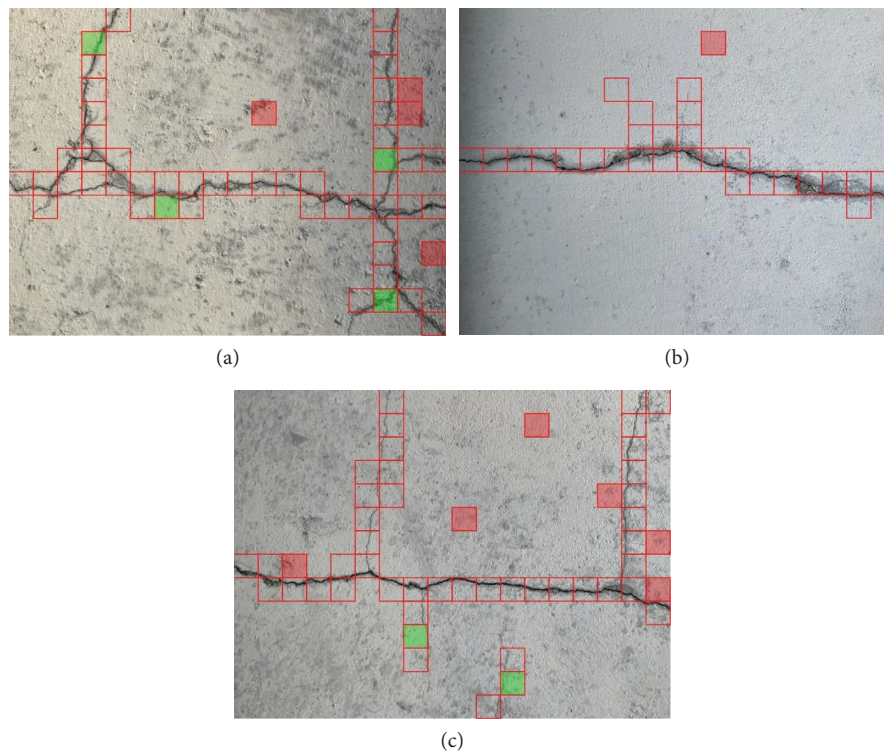


(c)

FIGURE 8: Crack detection results for three large-scale images. (a) Uneven lighting. (b) Hairline thin cracks. (c) Thin cracks.

TABLE 4: Crack detection accuracy for large-scale images.

Testing large-scale images	P	N	TP	TN	FP	FN	Accuracy (%)	Detection time (s)
a	49	203	45	199	4	4	96.83	0.92
b	27	225	27	224	1	0	99.60	0.88
c	44	208	42	202	6	2	96.83	0.91



(a)

(b)

(c)

FIGURE 9: Crack detection accuracy for three large-scale images (green shadowed: false negative, red shadowed: false positive).

another image, wherein pixel values indicate the distance to the nearest nonzero pixels. By aligning these values with the centerline pixels, the crack half-width along the crack length is determined. Consequently, the crack width is obtained by extracting distances along the skeleton images and then doubling the values. These geometric measurements of cracks

are expressed in terms of pixels. To establish physical units (e.g., mm), additional calibration objects or sensors are needed.

Overall, the framework for obtaining the final crack results involves the following steps: (1) The trained TL-LRDL was used to select the window containing cracks as shown in Figure 8. (2) Subsequently, the selected windows were

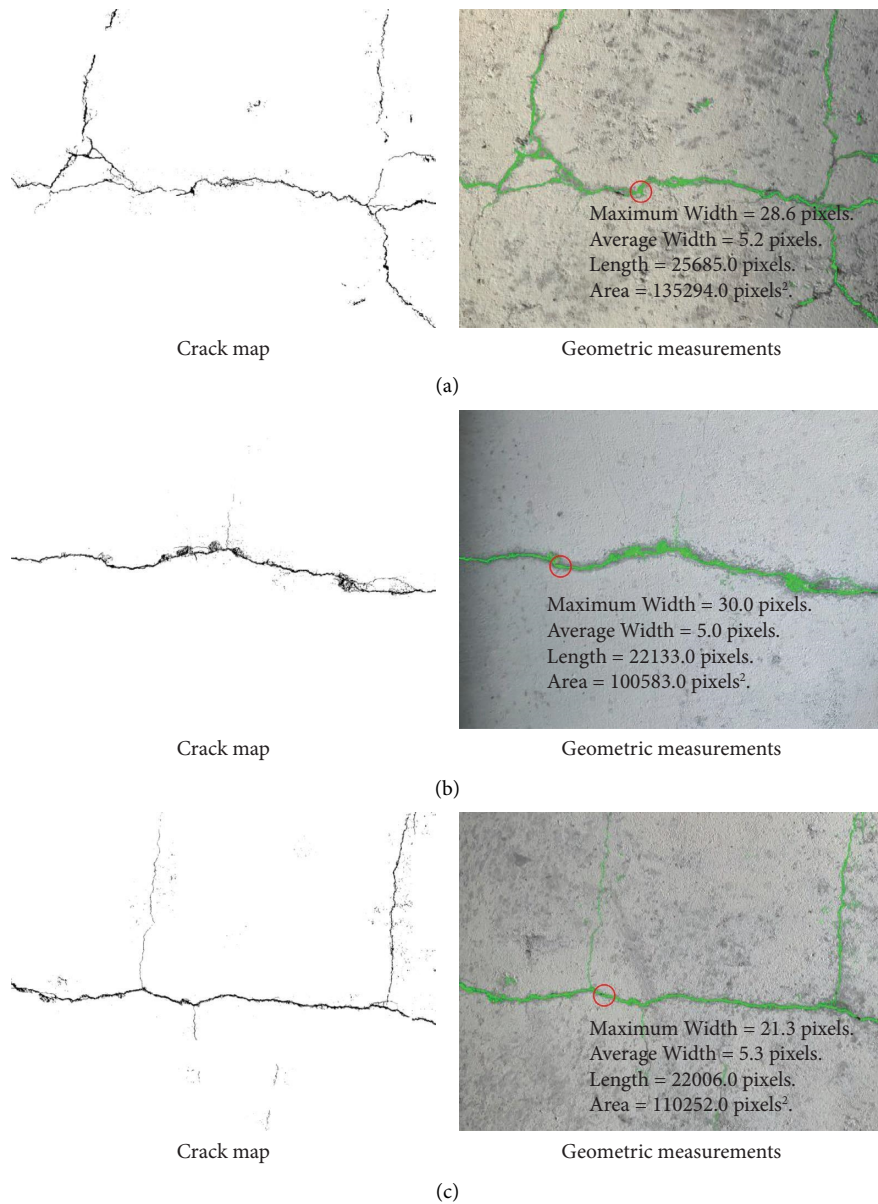


FIGURE 10: Crack pixel-level segmentation and geometric measurements for three large-scale images. (a) Uneven lighting. (b) Hairline thin cracks. (c) Thin cracks.

TABLE 5: Geometric measurements of cracks.

Testing large-scale image	Area (Pixel ²)	Length (Pixel)	Maximum width (Pixel)	Mean width (Pixel)
a	135294	25685	28.6	5.2
b	100583	22133	30	5
c	110252	22006	21.3	5.3

processed by Otsu's threshold and median filtering for pixel-level crack segmentation, and the crack maps were consequently obtained as shown in Figure 10 (left column). (3) Based on the binary crack maps, the crack geometric measurement was obtained using morphological operations, with results displayed in Figure 10 (right column).

It is important to note that the focus of this study is to develop a fast, easy implemented method for detecting cracks with a combination of TL and LRDL. The proposed TL-LRDL is essentially a classification method for crack detection. Therefore, the above steps 2 and 3 are only a subsequent task after crack detection, demonstrating that

the crack detection results of the proposed method can be further used for crack segmentation and quantification.

4. Comparative Study and Discussions

4.1. Comparison with ResNet18 under Different Training Modes. To provide insights into the efficiency and accuracy of the proposed TL-LRDL, it is compared with ResNet18 under two different training modes. In training mode 1, ResNet18 is trained from scratch without freezing any layers. In total, there are 11,182,338 trainable parameters. The hyperparameters are as follows: the initial learning rate of 0.1; weight decay and momentum values of 0.0001 and 0.9, respectively; the learning rate drop factor of 0.1; and the batch size of 100. The number of epochs is 20, and if there are two consecutive epochs without improvement, the learning rate will be reduced.

In training mode 2, ResNet18 is trained only around the final layer. All preceding layers remain fixed, with fine-tuning taking place only around the final layer. The initial weights in convolution layers are from the ImageNet and are not updated. The total number of network parameters is 11,182,338, and the number of trainable parameters is 1,026. For fair comparison, all tasks are performed on the same computer using the same training and testing sets.

Table 6 summarizes the comparative study results. Compared with ResNet18 under two training modes 1 and 2, the proposed method not only achieves better recognition accuracy but also requires significantly fewer trainable parameters. There are five main trainable parameters in TL-LRDL (\mathbf{Z} , \mathbf{Q} , \mathbf{R} , \mathbf{D} , \mathbf{E} as shown in equation (A.2)). During the dictionary learning process, they are continuously updated until convergence is achieved. Among them, the dictionary \mathbf{D} is the most important one.

The reduction in trainable parameters and model complexity leads to a faster training speed. The training time based on mode 1 and mode 2 is approximately 6 and 1 h, respectively, while the proposed TL-LRDL method requires only about 2 min 43 s. Excluding the time for feature extraction from ResNet18, the time for training a dictionary (LRDL) only took about 25.33 s. Experimental results demonstrate that the proposed TL-LRDL offers superior training efficiency and recognition accuracy, and its training requires no hardware acceleration, such as GPU. This is particularly suitable for edge computing, which involves processing data locally at the edge devices, rather than training the model at a centralized data center. It allows the proposed method to run effectively in an edge computing environment, on low-power devices, or close to where data are collected (e.g., low-powered UAVs, UGVs, and AR headsets with limited processing capabilities).

4.2. Comparison of Different Final Classifiers. In this section, different final classifier layers were selected to perform classification, while using the same pretrained network as feature extractor. The selected classifiers include support vector machine (SVM), K-nearest neighbor (KNN), single-layer neural network (NN), and LRDL. Pretrained ResNet18

was transferred and employed as feature extractor. For fair evaluation, each algorithm was trained and tested on identical datasets, with SVM and KNN parameters set to their default values. For the single-layer NN classifier, the “ReLU” activation was used in the hidden layer and “softmax” activation in the output layer. The specific results are presented in Table 7. Notably, the integration of ResNet18 and LRDL achieved better classification accuracy, followed by single-layer NN classifier.

4.3. Comparison of Different Pretrained Networks for Feature Extraction. In this section, different pretrained networks were selected to perform feature extraction while using the same LRDL classification model. The selected pretrained networks include VGG16, VGG19, GoogleNet, and ResNet18. Based on the transferred features from the pretrained networks, LRDL was used as the final classifier. The comparison results are presented in Table 8. From the table, it can be observed that ResNet18 and GoogleNet perform relatively well compared to VGG16. This is possibly because VGG16 has a simpler architecture, whereas ResNet has better image feature representation capabilities to capture the details and features in the images. Based on the experimental results in these two sections, it can be concluded that the proposed TL-LRDL method, which integrates ResNet18 and LRDL, is effective. This approach leverages the powerful feature extraction capabilities of ResNet18, and the efficient model training provided by LRDL.

4.4. Comparison with Traditional Edge Detection Methods. Furthermore, the proposed method is compared with two commonly used edge detectors: the Canny [62] and Sobel [63] edge detectors. These detectors have been widely used to detect the edges of cracks and serve as a standard baseline for comparing the performance of new crack detection methodologies in the literature [11, 26, 64–66]. Although some deep learning-based semantic segmentation methods (e.g., U-Net and FCN) could directly identify cracks at the pixel level across entire image, they are inherently supervised learning methods. Hence, pixel-wise labeling of samples is required, which is time-consuming. Instead, this study belongs to two-stage crack identification methods based on bounding box detection and image processing for segmentation.

Figure 11 presents a simple comparison of the proposed method, Canny and Sobel edge detectors on one of the large-scale images. As seen in Figure 11(b), the proposed method provides clear crack information. Compared to it, the Canny and Sobel edge detectors provide no meaningful information regarding cracks with high levels of noise, as shown in Figures 11(c) and 11(d). The noise scattered throughout the images could be attributed to the rough concrete surfaces and background noise. The comparison results show that the proposed method considerably outperforms the traditional Canny and Sobel methods.

4.5. Effects of the Number of Training Images. To demonstrate the advantage of the proposed method, which exhibits a reduced demand for training data, the effects of

TABLE 6: Comparison with classic DCNN methods.

Method	Trainable parameters	Training accuracy (%)	Testing accuracy (%)	Training time (s)
ResNet (train from scratch)	11,182,338	99.37	99.30	6 h 43 min 21
ResNet + FT	1,026	99.45	99.35	1 h 2 min 37
TL-LRLD (proposed)	5	99.71	99.68	2 min 43

TABLE 7: Experimental comparison under different final classifiers.

Feature extractor	Final classifier	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
ResNet18	SVM	97.19	98.88	95.66	97.24
	KNN	98.78	97.68	99.88	98.77
	NN	99.06	98.78	99.33	99.05
	LRDL	99.69	99.55	99.84	99.69

TABLE 8: Experimental comparison under different pretrained networks.

Feature extractor	Final classifier	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
VGG16	LRDL	98.50	97.49	99.48	98.47
VGG19		99.38	99.07	99.69	99.38
GoogleNet		99.63	99.57	99.70	99.63
ResNet18		99.68	99.93	99.42	99.67

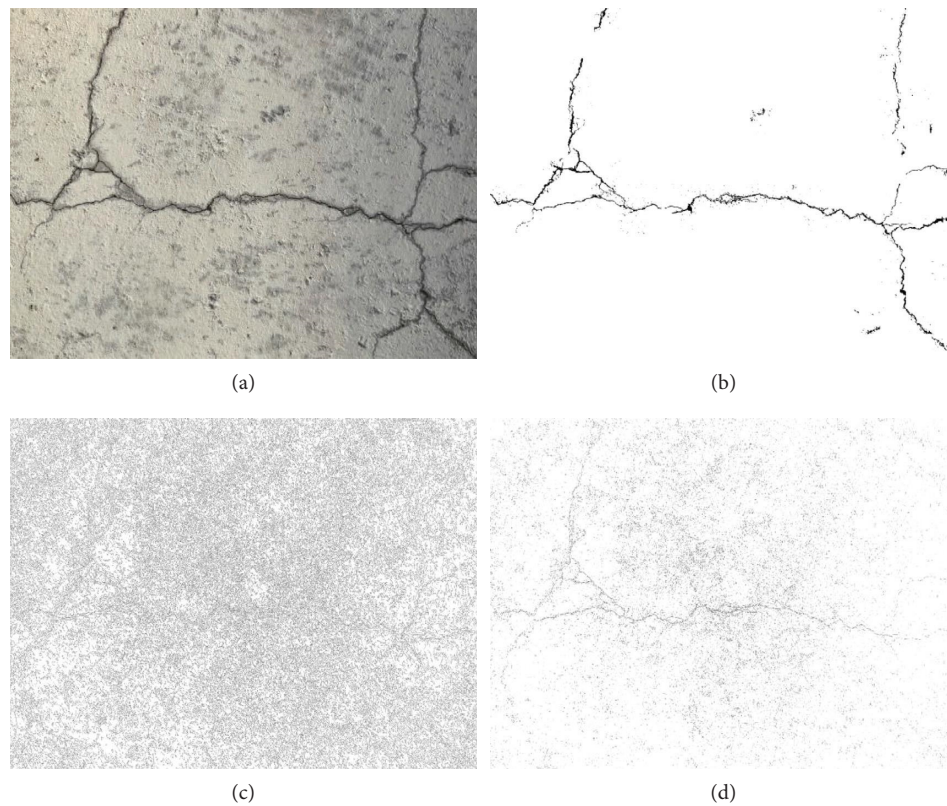


FIGURE 11: Comparisons with traditional edge detection methods: (a) original image; (b) output of the proposed method; (c) output of the Canny edge detector; (d) output of the Sobel edge detector.



FIGURE 12: Performance comparison under different number of images used for training.

the number of training images on classification performance are discussed. A consistent testing set, comprising 10,000 images, is used for the experiment. The number of training images is varied from 200 to 10,000 (Figure 12).

The results reveal two key observations. First, there is a positive correlation between the number of training images and the testing accuracy. However, the increase in accuracy is marginal as the training set size grows. This suggests that while having more training data is beneficial, the improvement in accuracy plateaus beyond a certain point in this study. Notably, when the training set is reduced to 1,000 images, we observe a marked decrease in performance. This finding underscores that a minimum of 1,000 training images is essential to effectively train the TL-LRDL algorithm and achieve a discriminative dictionary for accurate crack classification. Second, with the increase in the number of training images, the time for training a dictionary also increases. However, the maximum time for training is less than 1 min. This observation is consistent with previous findings, highlighting the high efficiency of the proposed method.

5. Conclusions

A novel crack detection method that comprises TL and LRDL, which enables the automated and fast crack detection, is presented in this study. Different from most of DCNN methods, the proposed method can greatly improve the training efficiency and reduce the demand for training data. Images from a public dataset were used for training and testing the proposed TL-LRDL model. The generated crack detection model was further implemented to detect the cracks on concrete surfaces under different scenarios. Through the comparison with other methods, the effectiveness and superiority of the proposed TL-LRDL method were successfully verified. Below are the major conclusions of this study:

- (1) The present study represents the first attempt to combine TL and LRDL in SHM for fast crack detection on concrete surfaces.
- (2) The proposed TL-LRDL method surmounts the obstacle for requiring long training time and high computational demand. In this study, the entire training process only took about 163.33 s (less than

3 min). Excluding the time taken for feature extraction from ResNet18, the time for training a dictionary (LRDL) was only about 25.33 s.

- (3) The proposed method reduces the demand for labeled data during training. Experimental results showed that the TL-LRDL achieved an accuracy of 99.68% with only 20% labeled data.
- (4) The generated TL-LRDL model can detect and locate cracks in large-scale images with high accuracy, even under uneven lighting, and exhibits a strong ability for detecting thin cracks, and even some hairline cracks could be successfully detected.

The above advantages facilitate the implementation of the proposed TL-LRDL on computers with limited resources, such as battery-powered UAVs, UGVs, and scarce processing capability of AR headsets. Furthermore, its fast-training process provides a potential solution to complete quick model updates when the dataset is updated. However, the current study only utilizes image data for crack detection. There exist several attempts to improve the crack detection performance by integrating other sensing data, e.g., infrared thermal sensing, ground-penetrating radar, and vibration data. Future studies will be carried out to extend the proposed algorithm to a fusion of multimodal data.

Appendix

A. Optimization Algorithm of equation (8)

The detailed optimization algorithm to solve the following optimization problem equation (A.1) is presented in this section:

$$\min_{\mathbf{Z}, \mathbf{D}, \mathbf{E}} \|\mathbf{Z}\|_* + \lambda \|\mathbf{E}\|_1 + \alpha \|\mathbf{W} \odot \mathbf{Z}\|_1 + \frac{\gamma}{2} \|\mathbf{D}\|_F^2 \quad (\text{A.1})$$

$$\text{s.t. } \mathbf{X} = \mathbf{D}\mathbf{Z} + \mathbf{E}.$$

Equation (A.1) can be solved based on the Augmented Lagrange Multiplier (ALM) method [67], and the Inexact Augmented Lagrange Multiplier (IALM) is adopted here for high efficiency. They fit for convex problem with separable objective functions and linear constraints. However, it is easily observed from equation (A.1) that the variable \mathbf{Z} is

subjected to multiple constraints, such as $\|\mathbf{Z}\|_*$ and $\|\mathbf{W} \odot \mathbf{Z}\|_1$. To make the problem easily solvable, it usually needs two auxiliary variables \mathbf{Q} and \mathbf{R} to decouple the objective function and constraints. Following Liu et al. [54], $\mathbf{Z} = \mathbf{Q}$ and $\mathbf{Z} = \mathbf{R}$ are adopted in equation (A.2) to make the objective function separable. Then, equation (A.1) can be converted to the following equivalent problem:

$$\min_{\mathbf{Z}, \mathbf{Q}, \mathbf{R}, \mathbf{D}, \mathbf{E}} \|\mathbf{Q}\|_* + \lambda \|\mathbf{E}\|_1 + \alpha \|\mathbf{W} \odot \mathbf{R}\|_1 + \frac{\gamma}{2} \|\mathbf{D}\|_F^2 \quad (\text{A.2})$$

$$\text{s.t. } \mathbf{X} = \mathbf{D}\mathbf{Z} + \mathbf{E}, \mathbf{Z} = \mathbf{Q}, \mathbf{Z} = \mathbf{R}.$$

Correspondingly, the augmented Lagrangian function for equation (A.2) is formulated as

$$\begin{aligned} \mathcal{L}(\mathbf{Z}, \mathbf{Q}, \mathbf{R}, \mathbf{D}, \mathbf{E}, \mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_3, \mu) = & \min_{\mathbf{Z}, \mathbf{Q}, \mathbf{R}, \mathbf{D}, \mathbf{E}} \|\mathbf{Q}\|_* + \lambda \|\mathbf{E}\|_1 + \alpha \|\mathbf{W} \odot \mathbf{R}\|_1 + \frac{\gamma}{2} \|\mathbf{D}\|_F^2 + \langle \mathbf{Y}_1, \mathbf{X} - \mathbf{D}\mathbf{Z} - \mathbf{E} \rangle \\ & + \langle \mathbf{Y}_2, \mathbf{Z} - \mathbf{Q} \rangle + \langle \mathbf{Y}_3, \mathbf{Z} - \mathbf{R} \rangle + \frac{\mu}{2} (\|\mathbf{X} - \mathbf{D}\mathbf{Z} - \mathbf{E}\|_F^2 + \|\mathbf{Z} - \mathbf{Q}\|_F^2 + \|\mathbf{Z} - \mathbf{R}\|_F^2), \end{aligned} \quad (\text{A.3})$$

where $\langle \mathbf{A}, \mathbf{B} \rangle = \text{trace}(\mathbf{A}^T \mathbf{B})$, $\mu > 0$ is a penalty parameter, and \mathbf{Y}_1 , \mathbf{Y}_2 , and \mathbf{Y}_3 are Lagrange multipliers. The optimization problem equation (A.3) can be solved iteratively by

separately updating the five optimization variables \mathbf{Q} , \mathbf{Z} , \mathbf{R} , \mathbf{E} , \mathbf{D} . The detailed updating scheme is as follows:

Updating \mathbf{Q} :

While fixing \mathbf{Z} , \mathbf{R} , \mathbf{E} , \mathbf{D} , \mathbf{Q} can be updated as follows:

$$\begin{aligned} \mathbf{Q}^{k+1} &= \arg \min_{\mathbf{Q}} \|\mathbf{Q}\|_* + \langle \mathbf{Y}_2^k, \mathbf{Z}^k - \mathbf{Q} \rangle + \frac{\mu^k}{2} \|\mathbf{Z}^k - \mathbf{Q}\|_F^2 \\ &= \arg \min_{\mathbf{Q}} \frac{1}{\mu^k} \|\mathbf{Q}\|_* + \frac{1}{2} \left\| \mathbf{Q} - \left(\mathbf{Z}^k + \frac{\mathbf{Y}_2^k}{\mu^k} \right) \right\|_F^2 = \mathbf{US}_{(1/\mu^k)}[\sum] \mathbf{V}^T, \end{aligned} \quad (\text{A.4})$$

where $(\mathbf{U}, \Sigma, \mathbf{V}^T) = \text{SVD}(\mathbf{Z}^k + (\mathbf{Y}_2^k/\mu^k))$ and $S_\varepsilon[\bullet]$ is the soft-thresholding operator defined as follows [54]:

$$S_\varepsilon[\bullet] = \begin{cases} x - \varepsilon, & \text{if } x > \varepsilon, \\ x + \varepsilon, & \text{if } x < -\varepsilon, \\ 0, & \text{otherwise.} \end{cases} \quad (\text{A.5})$$

Updating \mathbf{Z} :

While fixing \mathbf{Q} , \mathbf{R} , \mathbf{E} , \mathbf{D} , \mathbf{Z} can be updated as follows:

$$\begin{aligned} \mathbf{Z}^{k+1} &= \arg \min_{\mathbf{Z}} \langle \mathbf{Y}_1^k, \mathbf{X} - \mathbf{D}^k \mathbf{Z} - \mathbf{E} \rangle + \langle \mathbf{Y}_2^k, \mathbf{Z} - \mathbf{Q}^{k+1} \rangle \\ &+ \langle \mathbf{Y}_3^k, \mathbf{Z} - \mathbf{R}^k \rangle + \frac{\mu^k}{2} (\|\mathbf{X} - \mathbf{D}^k \mathbf{Z} - \mathbf{E}\|_F^2 + \|\mathbf{Z} - \mathbf{Q}^{k+1}\|_F^2 + \|\mathbf{Z} - \mathbf{R}^k\|_F^2). \end{aligned} \quad (\text{A.6})$$

The closed-form solution of equation (A.6) is as follows:

$$\mathbf{Z}^{k+1} = \left((\mathbf{D}^k)^T \mathbf{D}^k + 2\mathbf{I} \right)^{-1} \left[(\mathbf{D}^k)^T (\mathbf{X} - \mathbf{E}^k) + \mathbf{Q}^{k+1} + \mathbf{R}^k + \frac{(\mathbf{D}^k)^T \mathbf{Y}_1^k - \mathbf{Y}_2^k - \mathbf{Y}_3^k}{\mu^k} \right]. \quad (\text{A.7})$$

Updating \mathbf{R} :

While fixing $\mathbf{Q}, \mathbf{Z}, \mathbf{E}, \mathbf{D}$, \mathbf{R} can be updated as follows:

$$\begin{aligned}\mathbf{R}^{k+1} &= \arg \min_{\mathbf{R}} \alpha \|\mathbf{W} \odot \mathbf{R}\|_1 + \langle \mathbf{Y}_3^k, \mathbf{Z}^{k+1} - \mathbf{R} \rangle + \frac{\mu^k}{2} \left(\|\mathbf{Z}^{k+1} - \mathbf{R}\|_F^2 \right) \\ &= \arg \min_{\mathbf{R}} \alpha \|\mathbf{W} \odot \mathbf{R}\|_1 + \frac{\mu^k}{2} \left\| \mathbf{R} - \left(\mathbf{Z}^{k+1} + \frac{\mathbf{Y}_3^k}{\mu^k} \right) \right\|_F^2,\end{aligned}\tag{A.8}$$

where \mathbf{R}^{k+1} can be updated through the elementwise strategy. For each element \mathbf{R}_{ij} , the optimal solution of equation (A.8) can be computed as follows:

$$\mathbf{R}_{ij}^{k+1} = \arg \min_{\mathbf{R}_{ij}} \alpha \mathbf{W}_{ij} |\mathbf{R}_{ij}| + \frac{\mu^k}{2} \left\| \mathbf{R}_{ij} - \left(\mathbf{Z}_{ij}^{k+1} + \frac{(\mathbf{Y}_3^k)_{ij}}{\mu^k} \right) \right\|_F^2 = S_{(\alpha \mathbf{R}_{ij} / \mu^k)} \left(\mathbf{Z}_{ij}^{k+1} + \frac{(\mathbf{Y}_3^k)_{ij}}{\mu^k} \right).\tag{A.9}$$

Updating \mathbf{E} :

While fixing $\mathbf{Q}, \mathbf{Z}, \mathbf{R}, \mathbf{D}$, \mathbf{E} can be updated as follows:

$$\begin{aligned}\mathbf{E}^{k+1} &= \arg \min_{\mathbf{E}} \lambda \|\mathbf{E}\|_1 + \langle \mathbf{Y}_1^k, \mathbf{X} - \mathbf{D}^k \mathbf{Z}^{k+1} - \mathbf{E} \rangle + \frac{\mu^k}{2} \left(\|\mathbf{X} - \mathbf{D}^k \mathbf{Z}^{k+1} - \mathbf{E}\|_F^2 \right) \\ &= \arg \min_{\mathbf{E}} \frac{\lambda}{\mu^k} \|\mathbf{E}\|_1 + \frac{1}{2} \left\| \mathbf{E} - \left(\mathbf{X} - \mathbf{D}^k \mathbf{Z}^{k+1} + \frac{\mathbf{Y}_1^k}{\mu^k} \right) \right\|_F^2.\end{aligned}\tag{A.10}$$

Updating \mathbf{D} :

While fixing $\mathbf{Q}, \mathbf{Z}, \mathbf{R}, \mathbf{E}$, \mathbf{D} can be updated as follows:

$$\begin{aligned}\mathbf{D}^{k+1} &= \arg \min_{\mathbf{D}} \frac{\gamma}{2} \|\mathbf{D}\|_F^2 + \langle \mathbf{Y}_1^k, \mathbf{X} - \mathbf{D} \mathbf{Z}^{k+1} - \mathbf{E}^{k+1} \rangle + \frac{\mu^k}{2} \left\| \mathbf{X} - \mathbf{D} \mathbf{Z}^{k+1} - \mathbf{E}^{k+1} \right\|_F^2 \\ &= \left[\frac{\mathbf{Y}_1^k (\mathbf{Z}^{k+1})^T}{\mu^k} - (\mathbf{E}^{k+1} - \mathbf{X}) (\mathbf{Z}^{k+1})^T \right] \left(\frac{\gamma}{\mu^k} \mathbf{I} + \mathbf{Z}^{k+1} (\mathbf{Z}^{k+1})^T \right)^{-1}.\end{aligned}\tag{A.11}$$

The initial dictionary \mathbf{D}^0 is obtained by randomly selecting from training data.

The above alternative updating steps are repeated until the convergence condition is satisfied. The convergence condition of IALM is as follows: $\|\mathbf{Z}^{k+1} - \mathbf{Q}^{k+1}\|_\infty < \text{Tol}$,

$\|\mathbf{Z}^{k+1} - \mathbf{R}^{k+1}\|_\infty < \text{Tol}$, and $\|\mathbf{X} - \mathbf{D}^{k+1} \mathbf{Z}^{k+1} - \mathbf{E}^{k+1}\|_\infty < \text{Tol}$ are simultaneously satisfied, where $\text{Tol} > 0$ is a small tolerance value. The overall optimization process of solving equation (A.1) is summarized in Algorithm 1.

Input: Training data set \mathbf{X} ; Parameters λ, α, γ ; Weight matrix \mathbf{W}

- (1) Initialize $\mathbf{Z}^0 = 0, \mathbf{Q}^0 = 0, \mathbf{R}^0 = 0, \mathbf{E}^0 = 0, \mathbf{Y}_1^0 = 0, \mathbf{Y}_2^0 = 0, \mathbf{Y}_3^0 = 0, \mu = 10^{-6}, \mu_{\max} = 10^8, \text{Tol} = 10^{-6}, \rho = 1.15$
- (2) **While** not converged do
- (3) updating \mathbf{Q} by equation (A.4) while fixing the others
- (4) updating \mathbf{Z} by equation (A.7) while fixing the others
- (5) updating \mathbf{R} by equation (A.8) while fixing the others
- (6) updating \mathbf{E} by equation (A.10) while fixing the others
- (7) updating \mathbf{D} by equation (A.11) while fixing the others
- (8) updating the multipliers:
 $\mathbf{Y}_1^{k+1} = \mathbf{Y}_1^k + \mu^k (\mathbf{X} - \mathbf{D}^{k+1} \mathbf{Z}^{k+1} - \mathbf{E}^{k+1})$
 $\mathbf{Y}_2^{k+1} = \mathbf{Y}_2^k + \mu^k (\mathbf{Z}^{k+1} - \mathbf{W}^{k+1})$
 $\mathbf{Y}_3^{k+1} = \mathbf{Y}_3^k + \mu^k (\mathbf{Z}^{k+1} - \mathbf{R}^{k+1})$
- (9) updating μ :
 $\mu^{k+1} = \min(\mu_{\max}, \rho \mu^k)$
- (10) checking the convergence conditions
 $\|\mathbf{Z}^{k+1} - \mathbf{Q}^{k+1}\|_{\infty} < \text{Tol}, \|\mathbf{Z}^{k+1} - \mathbf{R}^{k+1}\|_{\infty} < \text{Tol},$ and
 $\|\mathbf{X} - \mathbf{D}^{k+1} \mathbf{Z}^{k+1} - \mathbf{E}^{k+1}\|_{\infty} < \text{Tol}$
- (11) **End While**

Output: $\mathbf{Z}_{\text{Train}}, \mathbf{D}$ and $\mathbf{E}_{\text{Train}}$

ALGORITHM 1: Solving equation (A.1) by IALM.

The major computational steps of Algorithm 1 are Steps 3 and 4, which require the singular value decomposition (SVD) of matrices. To accelerate the DL process, the training set is split into several batches, instead of simultaneously using the whole training dataset for DL. The batch size in this study is set as 200. The number of dictionary atoms is 100 ($m = 100$).

Data Availability

The data used to support the findings of this study are available from the authors upon reasonable request.

Conflicts of Interest

The authors declare that they have no known conflicts of financial interests or personal relationships that could have appeared to influence the work reported in this study.

Acknowledgments

This research has been supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region (SAR), China (Grant no. T22-501/23-R), and a grant from the Guangdong Basic and Applied Basic Research Foundation of Department of Science and Technology of Guangdong Province (Grant no. 2021B1515130006). The authors also appreciate the funding support by the Innovation and Technology Commission of Hong Kong SAR Government and the Hong Kong Branch of Chinese National Rail Transit Electrification and Automation Engineering Technology Research Centre (Grant no. K-BBY1).

References

- [1] J. Aboudi, "Stiffness reduction of cracked solids," *Engineering Fracture Mechanics*, vol. 26, no. 5, pp. 637–650, 1987.

- [2] J. H. Deng, A. Singh, Y. Y. Zhou, Y. Lu, and V. C. S. Lee, "Review on computer vision-based crack detection and quantification methodologies for civil structures," *Construction and Building Materials*, vol. 356, pp. 129238–129320, 2022.
- [3] A. J. Lee, W. Song, B. Yu, D. Choi, C. Tirtawardhana, and H. Myung, "Survey of robotics technologies for civil infrastructure inspection," *Journal of Infrastructure Intelligence and Resilience*, vol. 2, no. 1, pp. 100018–100112, 2023.
- [4] K. Zhou, Z. Wang, Y. Q. Ni, Y. Zhang, and J. Tang, "Unmanned aerial vehicle-based computer vision for structural vibration measurement and condition assessment: a concise survey," *Journal of Infrastructure Intelligence and Resilience*, vol. 2, no. 2, Article ID 100031, 2023.
- [5] A. Ghadimzadeh Alamdari and A. Ebrahimkhanlou, "A multi-scale robotic approach for precise crack measurement in concrete structures," *Automation in Construction*, vol. 158, Article ID 105215, 2024.
- [6] W. Q. Liu, E. Z. Rui, L. Yuan, S. Y. Chen, Y. L. Zheng, and Y. Q. Ni, "A novel computer vision-based vibration measurement and coarse-to-fine damage assessment method for truss bridges," *Smart Structures and Systems*, vol. 31, no. 4, pp. 393–407, 2023.
- [7] D. Ai, G. Jiang, S.-K. Lam, P. He, and C. Li, "Computer vision framework for crack detection of civil infrastructure - a review," *Engineering Applications of Artificial Intelligence*, vol. 117, Article ID 105478, 2023.
- [8] Y. Zhang and K. V. Yuen, "Review of artificial intelligence-based bridge damage detection," *Advances in Mechanical Engineering*, vol. 14, no. 9, Article ID 16878132221122770, 2022.
- [9] B. F. Spencer, V. Hoskere, and Y. Narazaki, "Advances in computer vision-based civil infrastructure inspection and monitoring," *Engineering*, vol. 5, no. 2, pp. 199–222, 2019.
- [10] I. Abdel-Qader, O. Abudayyeh, and M. E. Kelly, "Analysis of edge-detection techniques for crack identification in bridges," *Journal of Computing in Civil Engineering*, vol. 17, no. 4, pp. 255–263, 2003.

- [11] S. Dorafshan, R. J. Thomas, and M. Maguire, "Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete," *Construction and Building Materials*, vol. 186, pp. 1031–1045, 2018.
- [12] M. Kamaliardakani, L. Sun, and M. K. Ardakani, "Sealed-crack detection algorithm using heuristic thresholding approach," *Journal of Computing in Civil Engineering*, vol. 30, no. 1, Article ID 04014110, 2016.
- [13] Z. Qu, Y.-X. Chen, L. Liu, Y. Xie, and Q. Zhou, "The algorithm of concrete surface crack detection based on the genetic programming and percolation model," *IEEE Access*, vol. 7, pp. 57592–57603, 2019.
- [14] P. Bazrafshan, T. On, S. Basereh, P. Okumus, and A. Ebrahimkhanlou, "A graph-based method for quantifying crack patterns on reinforced concrete shear walls," *Computer-Aided Civil and Infrastructure Engineering*, vol. 39, no. 4, pp. 498–517, 2023.
- [15] M. Payab, R. Abbasina, and M. Khanzadi, "A brief review and a new graph-based image analysis for concrete crack quantification," *Archives of Computational Methods in Engineering*, vol. 26, no. 2, pp. 347–365, 2018.
- [16] T. Barisin, C. Jung, F. Müsebeck, C. Redenbach, and K. Schladitz, "Methods for segmenting cracks in 3D images of concrete: a comparison based on semi-synthetic images," *Pattern Recognition*, vol. 129, Article ID 108747, 2022.
- [17] R. X. Li, Y. C. Yuan, W. Zhang, and Y. L. Yuan, "Unified vision-based methodology for simultaneous concrete defect detection and geolocalization," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 7, pp. 527–544, 2018.
- [18] Y. Xu, S. L. Li, D. Y. Zhang et al., "Identification framework for cracks on a steel structure surface by a restricted Boltzmann machines algorithm based on consumer-grade camera images," *Structural Control and Health Monitoring*, vol. 25, no. 2, pp. 20755–e2120, 2018.
- [19] M. R. Jahanshahi, S. F. Masri, C. W. Padgett, and G. S. Sukhatme, "An innovative methodology for detection and quantification of cracks through incorporation of depth perception," *Machine Vision and Applications*, vol. 24, no. 2, pp. 227–241, 2013.
- [20] M. R. Jahanshahi and S. F. Masri, "A new methodology for non-contact accurate crack width measurement through photogrammetry for automated structural safety evaluation," *Smart Materials and Structures*, vol. 22, no. 3, pp. 035019–35112, 2013.
- [21] E. Zalama, J. Gómez-García-Bermejo, R. Medina, and J. Llamas, "Road crack detection using visual features extracted by Gabor filters," *Computer-Aided Civil and Infrastructure Engineering*, vol. 29, no. 5, pp. 342–358, 2014.
- [22] F. C. Chen, M. R. Jahanshahi, R. T. Wu, and C. Joffe, "A texture-based video processing methodology using bayesian data fusion for autonomous crack detection on metallic surfaces," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 4, pp. 271–287, 2017.
- [23] L. Nanni, S. Ghidoni, and S. Brahmam, "Handcrafted vs. non-handcrafted features for computer vision classification," *Pattern Recognition*, vol. 71, pp. 158–172, 2017.
- [24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
- [25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, <http://arxiv.org/abs/14091556>.
- [26] Y. J. Cha, W. Choi, and O. Büyüköztürk, "Deep learning-based crack damage detection using convolutional neural networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, pp. 361–378, 2017.
- [27] F. C. Chen and M. R. Jahanshahi, "NB-CNN: deep learning-based crack detection using convolutional neural network and naïve Bayes data fusion," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 5, pp. 4392–4400, 2018.
- [28] K. Gopalakrishnan, S. K. Khaitan, A. Choudhary, and A. Agrawal, "Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection," *Construction and Building Materials*, vol. 157, pp. 322–330, 2017.
- [29] K. Gopalakrishnan, H. Gholami, A. Vidyadharan, A. Choudhary, and A. Agrawal, "Crack damage detection in unmanned aerial vehicle images of civil infrastructure using pre-trained deep learning model," *International Journal for Traffic and Transport Engineering*, vol. 8, no. 1, pp. 1–14, 2018.
- [30] Y. Zhang and K. V. Yuen, "Crack detection using fusion features-based broad learning system and image processing," *Computer-Aided Civil and Infrastructure Engineering*, vol. 36, no. 12, pp. 1568–1584, 2021.
- [31] P. Palevicius, M. Pal, M. Landauskas, U. Orinaite, I. Timofejeva, and M. Ragulskis, "Automatic detection of cracks on concrete surfaces in the presence of shadows," *Sensors*, vol. 22, no. 10, p. 3662, 2022.
- [32] U. Orinaite, V. Karaliute, M. Pal, and M. Ragulskis, "Detecting underwater concrete cracks with machine learning: a clear vision of a murky problem," *Applied Sciences*, vol. 13, no. 12, p. 7335, 2023.
- [33] F. T. Ni, J. Zhang, and Z. Q. Chen, "Zernike-moment measurement of thin-crack width in images enabled by dual-scale deep learning," *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 5, pp. 367–384, 2018.
- [34] C. V. Dung, H. Sekiya, S. Hirano, T. Okatani, and C. Miki, "A vision-based method for crack detection in gusset plate welded joints of steel bridges using deep convolutional neural networks," *Automation in Construction*, vol. 102, pp. 217–229, 2019.
- [35] Y. J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, and O. Büyüköztürk, "Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 9, pp. 731–747, 2018.
- [36] J. H. Deng, Y. Lu, and V. C. S. Lee, "Concrete crack detection with handwriting script interferences using faster region-based convolutional neural network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 35, no. 4, pp. 373–388, 2020.
- [37] S. Jiang and J. Zhang, "Real-time crack assessment using deep neural networks with wall-climbing unmanned aerial system," *Computer-Aided Civil and Infrastructure Engineering*, vol. 35, no. 6, pp. 549–564, 2019.
- [38] D. Zou, M. Zhang, Z. Bai et al., "Multicategory damage detection and safety assessment of post-earthquake reinforced concrete structures using deep learning," *Computer-Aided Civil and Infrastructure Engineering*, vol. 37, no. 9, pp. 1188–1204, 2022.
- [39] Q. Qiu and D. Lau, "Real-time detection of cracks in tiled sidewalks using YOLO-based method applied to unmanned aerial vehicle (UAV) images," *Automation in Construction*, vol. 147, Article ID 104745, 2023.
- [40] A. Zhang, K. C. P. Wang, B. Li et al., "Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 10, pp. 805–819, 2017.

- [41] A. Zhang, K. C. P. Wang, Y. Fei et al., "Automated pixel-level pavement crack detection on 3D asphalt surfaces with a recurrent neural network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 3, pp. 213–229, 2019.
- [42] C. V. Dung and L. D. Anh, "Autonomous concrete crack detection using deep fully convolutional neural network," *Automation in Construction*, vol. 99, pp. 52–58, 2019.
- [43] T. Chen, Z. Cai, X. Zhao et al., "Pavement crack detection and recognition using the architecture of segNet," *Journal of Industrial Information Integration*, vol. 18, Article ID 100144, 2020.
- [44] S. Bhowmick, S. Nagarajaiah, and A. Veeraraghavan, "Vision and deep learning-based algorithms to detect and quantify cracks on concrete surfaces from uav videos," *Sensors*, vol. 20, no. 21, pp. 6299–6319, 2020.
- [45] L. Zhang, J. Shen, and B. Zhu, "A research on an improved Unet-based concrete crack detection algorithm," *Structural Health Monitoring*, vol. 20, no. 4, pp. 1864–1879, 2021.
- [46] W. Lu, M. Qian, Y. Xia et al., "Crack _ PSTU: crack detection based on the U-Net framework combined with Swin Transformer," *Structures*, vol. 62, Article ID 106241, 2024.
- [47] M. Pal, P. Palevičius, M. Landauskas, U. Orinaitė, I. Timofejeva, and M. Ragulskis, "An overview of challenges associated with automatic detection of concrete cracks in the presence of shadows," *Applied Sciences*, vol. 11, no. 23, Article ID 11396, 2021.
- [48] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, Las Vegas, NV, USA, June 2016.
- [49] O. Ronneberger, P. Fischer, and T. Brox, "U-net: convolutional networks for biomedical image segmentation," in *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, Munich, Germany, October 2015.
- [50] J. Shi, J. Dang, M. Cui et al., "Improvement of damage segmentation based on pixel-level data balance using VGG-Unet," *Applied Sciences*, vol. 11, no. 2, p. 518, 2021.
- [51] K. M. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.
- [52] Y. Q. Gao and K. M. Mosalam, "Deep transfer learning for image-based structural damage recognition," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 9, pp. 748–768, 2018.
- [53] G. Gao, J. Yang, X. Y. Jing, F. Shen, W. Yang, and D. Yue, "Learning robust and discriminative low-rank representations for face recognition with occlusion," *Pattern Recognition*, vol. 66, pp. 129–143, 2017.
- [54] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 171–184, 2013.
- [55] S. Y. Chen, Y. W. Wang, and Y. Q. Ni, "Gross outlier removal and fault data recovery for SHM data of dynamic responses by an annihilating filter-based Hankel-structured robust PCA method," *Structural Control and Health Monitoring*, vol. 29, no. 12, Article ID e3144, 2022.
- [56] R. Rubinfeld, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.
- [57] Y. Li, Y. Chai, H. Zhou, and H. Yin, "A novel dimension reduction and dictionary learning framework for high-dimensional data classification," *Pattern Recognition*, vol. 112, Article ID 107793, 2021.
- [58] S. Zhao, J. Wu, B. Zhang, and L. Fei, "Low-rank inter-class sparsity based semi-flexible target least squares regression for feature representation," *Pattern Recognition*, vol. 123, Article ID 108346, 2022.
- [59] Y. Zhang, Z. Jiang, and L. S. Davis, "Learning structured low-rank representations for image classification," in *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 676–683, Portland, OR, USA, June 2013.
- [60] Ç.F. Özgenel, "Concrete crack images for classification," *Mendeley Data*, 2018.
- [61] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.
- [62] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 679–698, 1986.
- [63] N. D. Hoang and Q. L. Nguyen, "Metaheuristic optimized edge detection for recognition of concrete wall cracks: a comparative study on the performances of roberts, prewitt, canny, and sobel algorithms," *Advances in Civil Engineering*, vol. 2018, Article ID 7163580, 16 pages, 2018.
- [64] Q. Y. Zhang, K. Barri, S. K. Babanajad, and A. H. Alavi, "Real-time detection of cracks on concrete bridge decks using deep learning in the frequency domain," *Engineering*, vol. 7, no. 12, pp. 1786–1796, 2021.
- [65] M. Alipour and D. K. Harris, "Increasing the robustness of material-specific deep learning models for crack detection across different materials," *Engineering Structures*, vol. 206, pp. 110157–110214, 2020.
- [66] F. T. Ni, J. Zhang, and Z. Q. Chen, "Pixel-level crack delineation in images with convolutional feature fusion," *Structural Control and Health Monitoring*, vol. 26, no. 1, pp. 22866–e2318, 2019.
- [67] Z. Lin, M. Chen, and Y. Ma, "The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices," 2010, <https://arxiv.org/abs/1009.5055>.