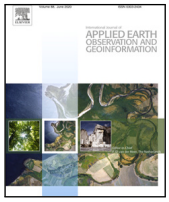
International Journal of Applied Earth Observation and Geoinformation 118 (2023) 103248

Contents lists available at ScienceDirect

International Journal of Applied Earth Observation and Geoinformation

journal homepage: www.elsevier.com/locate/jag



SCL-GCN: Stratified Contrastive Learning Graph Convolution Network for pavement crack detection from mobile LiDAR point clouds

Huifang Feng [a], Lingfei Ma [b,*], Yongtao Yu [c], Yiping Chen [d,*], Jonathan Li [e,f]

[a] Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Informatics, Xiamen University, Xiamen 361005, China
[b] School of Statistics and Mathematics, Central University of Finance and Economics, Beijing 102206, China
[c] Faculty of Computer and Software Engineering, Huaiyin Institute of Technology, Huaian 223003, China
[d] School of Geospatial Engineering and Science, Sun Yat-sen University, 519082 Zhuhai, China
[e] Department of Geography and Environmental Management, University of Waterloo, Waterloo, ON N2L 3G1, Canada
[f] Department of Systems Design Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada

ARTICLE INFO

Keywords:
Point clouds
Pavement crack detection
Stratify strategy
Contrastive learning
Graph convolution

ABSTRACT

Accurate pavement crack detection is important for routine maintenance of pavements and reduction of possible traffic accidents. Most existing rule- or learning-based point-level approaches cannot achieve high detection accuracy and efficiency owing to the disorderly arrangement, scattered intensities, diverse crack structures, large data volumes, and complex annotation of mobile laser scanning (MLS) point clouds. To address these issues, we developed SCL-GCN, a Stratified Contrastive Learning Graph Convolution Network with a novel dual-branch architecture for MLS-point cloud-based pavement crack detection. First, a multi-scale graph representation construction module was designed based on a stratification strategy. This module creates strengthened spaces for the raw pavement point cloud and its downsampled subset, from which adjacency matrices and initial representations are generated. The stratification strategy samples neighbors densely in the raw point clouds and sparsely in the downsampled subset to form the neighborhood for each point, utilizing long-range contexts to increase the effective receptive field while lowing the extra computation. Next, a graph feature contrastive learning module is proposed to take advantage of stratified features. This module supervises the learning process of the two branches to avoid learning bias caused by an imbalanced data distribution, promoting convergence and improving performance. The experimental results show that the developed SCL-GCN model outperforms state-of-the-art methods. With a training/testing ratio of only 1:6 and an overall training time of less than 70 min, the average precision, recall, and F_1-*score* of the SCL-GCN reached 75.7%, 75.1%, and 75.2%, respectively.

1. Introduction

Pavement cracking decreases the load-bearing capacity and water resistance of pavements, accelerates the deterioration process of roads, and poses a safety hazard to road users (Salman et al., 2013). Transverse, longitudinal, and crocodile cracks are the three primary forms of pavement cracking (Cubero-Fernandez et al., 2017). These are mainly caused by excessive and repetitive stress, moisture, adverse environmental conditions (such as freezing and thawing), and poor construction quality (Liu et al., 2020). The constant traffic flow in urban areas further exacerbates the formation of pavement cracks, as road quality deteriorates over time owing to wear and tear. Pavements require regular inspection, assessment, maintenance, and repair to ensure safety and extend their service life. Thus, one of the most important tasks to be conducted is detection of cracks in pavements (Hou et al., 2021).

Pavement crack detection is generally based on manual inspection and external sensors that utilize a variety of technologies, including acoustic emission (Ohno and Ohtsu, 2010), infrared thermography (Dabous et al., 2017), and ground-penetrating radar (Hong et al., 2017). Manual visual inspection is still one of the most common methods of pavement crack detection owing to its high operability (Kang et al., 2020). However, its low efficiency, high cost, intense subjectivity, susceptibility to vehicular traffic, and specialized domain knowledge make large-scale crack detection on road networks complex (Liu et al., 2019). Other methods based on embedded or external sensors are vulnerable to adverse environmental conditions such as changes in temperature and humidity (Kang et al., 2020). Thus, there is an urgent

* Corresponding authors.
E-mail addresses: l53ma@cufe.edu.cn (L. Ma), chenyp79@mail.sysu.edu.cn (Y. Chen).

https://doi.org/10.1016/j.jag.2023.103248
Received 23 December 2022; Received in revised form 13 February 2023; Accepted 23 February 2023
Available online 14 March 2023

need for an automated, efficient, non-contact approach for detecting cracks (Liu et al., 2022).

Traditional and machine-learning computer vision algorithms have been extensively studied for automatic detection of pavement cracks. Traditional methods based on image processing fall into three main categories: threshold segmentation (Oliveira and Correia, 2009; Peng et al., 2015), edge detection (Lim et al., 2011, 2014), and region growth (Gavilán et al., 2011; Zou et al., 2012). However, these methods generally rely on prior knowledge or optimal thresholds, and are poorly applicable to complex and changing urban roads. Such methods make it difficult to reliably and precisely detect cracks with weak connections or uneven geometric topologies.

The advent of machine learning, particularly deep learning, has taken crack detection to a new level, learning high-level features and significantly improving the quality of crack detection (Tong et al., 2020). However, these image-based methods still struggle to provide accurate results for pavement cracks because pavement images are often obscured by light, shadows, dirt, and noise, making it difficult to accurately capture the topography and texture details of the road.

With the development of three-dimensional (3D) data acquisition techniques, MLS systems have been widely used to produce data that contain accurate and reliable 3D coordinates, enabling efficient and flexible pavement point cloud acquisition. Accordingly, several characteristics of pavement point clouds are revealed: (1) the elevation and intensity of cracks are usually lower than those of the surrounding pavement points; (2) the point distribution and physical shape of cracks are very irregular; and (3) the pavement is typically located on flat terrain with equal or continuous elevation changes. Nevertheless, the elevation and intensity variations between cracks and normal pavement are not easily distinguishable owing to the effects of road wear and pavement texture noise, making the detection of pavement crack points challenging.

There are three limitations in most existing point-cloud-based crack detection methods: (1) disorganized point clouds cannot be simply processed using algorithms designed for regular grid structures; most studies require dimensionality or resolution reduction to transform them into images or voxel structures, which invariably causes information loss; (2) due to the flattened structure of pavements, existing point-based methods tend to ignore the spatial correlation and adjacency between adjacent points, resulting in incomplete crack detection for complex and large pavements; (3) point-based deep learning models rely heavily on the amount and quality of manually annotated data, require many parameters, and increase the computational cost as the network complexity increases. Consequently, the applicability of these approaches to different scenarios is limited.

In our previous study (Feng et al., 2022), CrackGCN proposed a solution that combined a space-strengthened graph representation with fine-grained contextual features. This was the first attempt to combine a graph convolutional network (GCN) with point-based crack detection to achieve superior performance with high efficiency and low data dependency. However, CrackGCN only considers fixed-size neighborhoods when constructing point features, ignoring the multi-scale information contained in multi-scale neighborhoods. This leads to incomplete results when CrackGCN addresses complex pavement cracks. To address this issue, this study presents SCL-GCN, a Stratified Contrastive Learning Graph Convolution Network. The SCL-GCN adopts a stratified strategy for multi-scale feature construction, constructs a dual-branch GCN architecture for multi-scale feature learning, and supervises the two-branch learning process with a contrastive learning mechanism. The main contributions of the SCL-GCN are summarized as follows:

(1) We propose a novel dual-branch architecture based on a GCN for semi-supervised detection of pavement cracks from MLS data.

(2) We adopt a unique stratification strategy to explore the multi-resolution local relative position information and construct multi-scale features of the pavement point cloud, helping to expand the effective reception field and increase the detection reliability of pavement cracks.

(3) We apply contrastive learning to supervise the training process of the two branches, avoid learning bias caused by imbalanced data distribution, and explore deeper features, promoting convergence and improving the performance of crack detection.

2. Related work

2.1. Image-based methods

Traditional methods. Pavement cracks in images appear as irregular stripes, making them difficult to detect due to their varying intensities, complex topologies, and low contrast or noisy backgrounds. Various image processing techniques including threshold segmentation (Oliveira and Correia, 2009; Peng et al., 2015), edge detection (Lim et al., 2011, 2014), and region growing (Gavilán et al., 2011; Zou et al., 2012) have been successfully applied to pavement crack-detection. Threshold segmentation-based methods separate cracks from the background by setting an appropriate pixel intensity threshold and dividing the image pixels into classes. Oliveira and Correia (2009) proposed an automatic crack-detection and classification framework that benefited from dynamic thresholds to identify potential dark pixels and generated the entropy block matrix used for crack-pixel identification. Peng et al. (2015) developed a crack-detection approach consisting of a reformative Otsu thresholding algorithm and an improved adaptive iterative threshold algorithm. Edge detection-based methods detect the edges of cracks by using edge detection operators. Lim et al. (2011, 2014) adopted the Laplacian of the Gaussian algorithm to detect cracks in high-resolution bridge deck images. Region-growing-based methods represent particular information within cracks by combining pixels with comparable features. Gavilán et al. (2011) designed a seed-based region growing algorithm that combined multi-directional non-minimal suppression with symmetry checking for road crack detection. CrackTree derived a minimal spanning tree by selecting crack seeds in a probability map and performing recursive tree edge pruning to detect the desired cracks (Zou et al., 2012). However, for these methods, proper parameter presetting or prior knowledge is critical to getting optimal performance without manual intervention, making them unsuitable for complex and changing urban roads. Pavement cracks with limited connectivity or irregular geometric topology are challenging to identify correctly and robustly.

Deep learning methods. In recent years, deep learning models have demonstrated impressive performance in terms of efficiency and accuracy in road hazard assessment owing to their excellent feature learning capabilities. Cha et al. (2017) proposed a deep learning model based on a CNN for detecting cracks in concrete surfaces without the use of image processing techniques. To identify pavement cracks, Tong et al. (2020) used a non-destructive testing technology based on ground-penetrating radar and network in networks. Yang et al. (2020) proposed a feature pyramid and a hierarchical boosting network. Although the impressive results have been obtained using these image-based learning methods, their performance is highly dependent on external conditions; in other words, their high sensitivity to shadows, noise, and stains adversely limits detection accuracy.

2.2. Point cloud-based methods

Traditional methods. Advances in 3D sensors have produced pavement point clouds that provide the location and shape of cracks and are insensitive to changes in lighting or weather conditions. Research on crack detection using point clouds has gained momentum. Guan et al. (2015) proposed ITVCrack, which use the inverse distance weighted (IDW) algorithm, iterative tensor voting, and morphological refinement to distinguish crack curves in MLS pavements. Yu et al. (2014) utilized

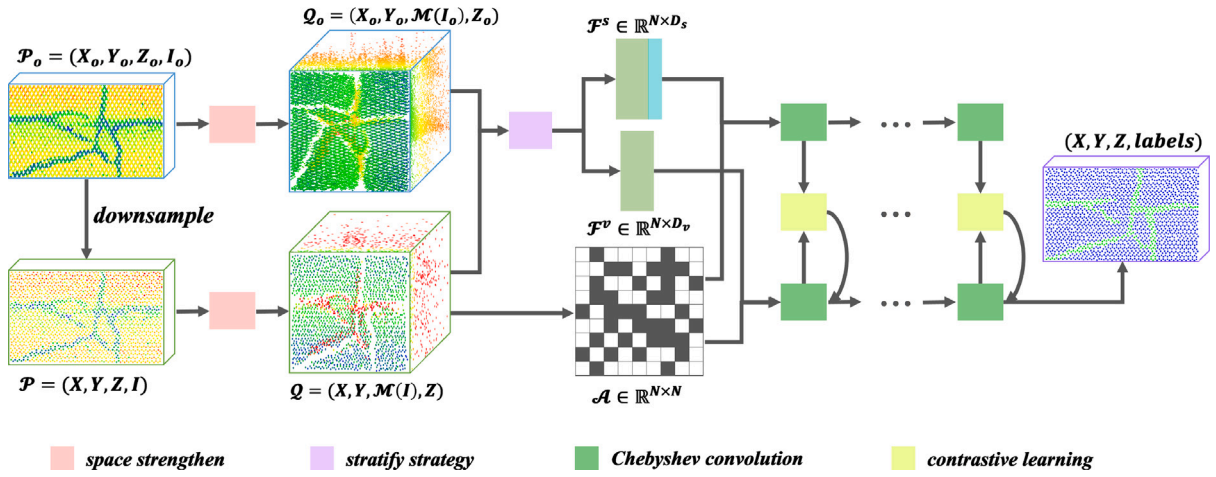


Fig. 1. Framework of the proposed SCL-GCN.

Otsu thresholding and Euclidean-based clustering to efficiently identify the crack skeletons in MLS point clouds. Jiang et al. (2018) adopted the plane triangulation modeling method to detect crack points for a triangular irregular network dataset built using the IDW rasterization method. Xu and Yang (2019) introduced a Gaussian filter to detect cracks with the largest SNR distribution gradient, effectively optimizing pavement damage analysis accuracy. However, these approaches struggle with detection of fine cracks or cracks with low connectivity and suffer from low automation.

Deep learning methods. Deep learning has proven to be powerful for 3D pattern recognition problems and has been successfully applied to detect cracks in 3D pavements. Turkan et al. (2018) created an adaptable wavelet neural network model to automatically detect concrete cracks; however, errors occurred with fine and shallow cracks. Zhang et al. (2017) developed a CNN architecture, CrackNet, consisting of five convolutional and fully connected layers with over one million learnable parameters. CrackNet achieved high pixel accuracy but was inefficient, and the static, non-learnable feature generator limited its learning capability. Several CrackNet-based studied aimed at improving learning capacity and efficiency were reported. CrackNet II (Zhang et al., 2018) abandoned the feature generator in favor of a more complex framework. CrackNet-V (Fei et al., 2020) adopted a smaller filter and proposed a new shallow crack activation unit. Although they outperform CrackNet in terms of efficiency and accuracy, the improvements are limited and highly data-dependent. Furthermore, the pixel-level results produced by these CrackNet-based models lack detailed information on crack locations.

GCN-based methods. Some researchers have turned to graph convolutions for better feature representations and to explore architectures that can achieve complete and precise crack detection results. Feng et al. (2022) developed CrackGCN, using a novel space strengthening strategy and fine-designed graph-based features to detect crack points from MLS data, achieving outstanding performance with high efficiency and low data dependency. Ma and Li (2022) proposed SD-GCN, a saliency-based dilated GCN architecture that uses two saliency feature spaces and cylinder-based dilated graph convolutions to extract cracks from the MLS data. Both CrackGCN and SD-GCN benefited from spatial augmentation strategies to amplify the geometric structures of pavement point clouds; however, they failed to consider long-range neighborhoods and multi-scale features, resulting in incomplete results with cracks in complex structures.

3. Method

Based on the assumption that connected nodes in the graph may share the same label, we constructed a pavement point cloud containing only some of the available labels as an undirected graph. For the input pavement point cloud $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$, where N is the number of points in $\mathcal{P}$, each containing its spatial coordinates and intensity information. For point $p_i \in \mathcal{P}$, $i \in [1, N]$, and $p_i = (x_i, y_i, z_i, I_i)$. The corresponding raw pavement point cloud of $\mathcal{P}$ is $\mathcal{P}_o$, where $\mathcal{P}$ is the downsampled subset of $\mathcal{P}_o$. Subsequently, we presented SCL-GCN, a stratified contrastive learning graph convolution network to transform the pavement crack detection task into a graph-based semi-supervised binary classification problem, for pavement crack detection from MLS point clouds. The SCL-GCN is illustrated in Fig. 1.

3.1. Overview

The proposed SCL-GCN consists of two modules: (1) a multi-scale graph representation construction module, and (2) a graph feature contrastive learning module. In the first module, we use the features constructed in CrackGCN (Feng et al., 2022) as the vanilla representation, and then obtain the stratified representation using a novel pointwise stratification strategy on this basis. In the second module, we designed a dual-branch graph convolution architecture to learn the features of the vanilla and stratified versions separately. Then, we used contrastive learning to supervise the dual-branch training process and obtain an accurate and complete pavement crack detection result.

3.2. Multi-scale graph representation construction

This module consists of two progressive steps, aimed at completing the adjacency relationship and feature construction of $\mathcal{P}$ and $\mathcal{P}_o$. In the first step, the adjacent relationships of the points in $\mathcal{P}$ and $\mathcal{P}_o$ are determined separately during pavement space strengthening, which can be represented by adjacency matrices $\mathcal{A}$ and $\mathcal{A}_o$, respectively. The second step is to construct multi-scale features, including vanilla and stratified features, for the input pavement point clouds $\mathcal{P}$. The vanilla feature is denoted as $\mathcal{F}^v \in \mathbb{R}^{N \times D_v}$ and the stratified feature as $\mathcal{F}^s \in \mathbb{R}^{N \times D_s}$, where D_v and D_s denote the feature channels.

3.2.1. Pavement space strengthening

Due to the low distinguishability in the spatial geometry and topological structure of pavement point clouds, we used the space strengthening-strategy proposed by Feng et al. (2022) to map the point clouds into the strengthened feature space, enhance the compactness within classes, and mitigate the imbalance between classes according to the intensity of the pavement point clouds. During space strengthening, $\mathcal{P}$ and $\mathcal{P}_o$ are converted into strengthened feature spaces $\mathcal{Q}$ and $\mathcal{Q}_o$, respectively. For example, for $\forall p_i \in \mathcal{P}$, $p_i = (x_i, y_i, z_i, I_i)$, which is

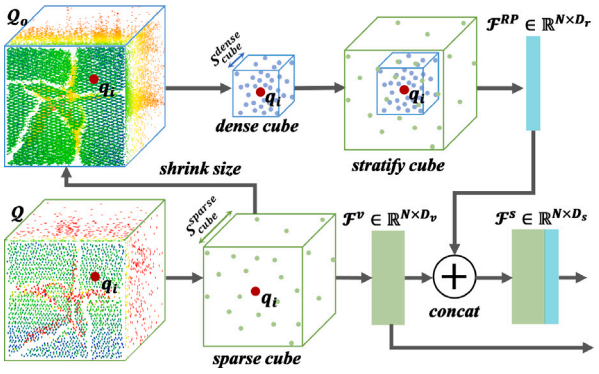


Fig. 2. The proposed stratify strategy.

transformed into $q_i \in \mathcal{Q}$, where $q_i = \left(x_i, y_i, \mathcal{M}(I_i), z_i\right)$. The formula for $\mathcal{M}(\cdot)$ is:

$$\mathcal{M}(I_i) = \csc\left(\frac{1}{1+e^{I_i}}\right) \tag{1}$$

where $csc(\cdot)$ is the cosecant function.

A fixed-radius nearest neighbor (fr-NN) search algorithm with radius R was used to determine the neighborhood of each point in the corresponding strengthened space. The space-strengthened adjacency matrix of $\mathcal{Q}$ is denoted as $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_N\}$. Specifically, for $\mathcal{A}_i \in \mathcal{A}$, $\mathcal{A}_i = \{a_{i1}, \ldots, a_{in}\}$, where $a_{ij} \in \mathcal{A}_i$ represents the adjacent relationship of points $q_i, q_j \in \mathcal{Q}$, which can be calculated as:

$$a_{ij} = \begin{cases} 1, & \left\|q_i, q_j\right\|_2 \le R \\ 0, & else \end{cases} \tag{2}$$

where $\|\cdot\|_2$ calculates the Euclidean distance between q_i and q_j, with $a_{ij} = 1$ indicating that q_i and q_j are adjacent to each other. Accordingly, the space-strengthened adjacency matrix of $\mathcal{Q}_o$ is denoted as $\mathcal{A}_o$, which can be calculated in the same manner as $\mathcal{A}$.

3.2.2. Multi-scale feature construction

The features generated from the vanilla and stratified versions construct multi-scale features. The vanilla version focuses on the local contextual information for each point in the input pavement point cloud. As 3D points have a more complex discrete structure than 2D regular pixels, the determination of XYZ positions is challenging. Thus, the stratified version, aims to explore features over long distances while better using each point's spatial position.

Vanilla features. We follow the feature construction strategy of Feng et al. (2022) to obtain the vanilla features $\mathcal{F}^v$ in $\mathcal{Q}$. Denoting the feature matrix of $\mathcal{Q}$ as $\mathcal{F}^v = \{\mathcal{F}^v_1, \ldots \mathcal{F}^v_N\}$, $\mathcal{F}^v_i \in \mathbb{R}^{1\times D_v}$ represents the feature vector of $q_i \in \mathcal{Q}$. According to Feng et al. (2022), $\mathcal{F}^v_i$ consists of 16 channels ($D_v = 16$), which can be expressed as:

$$\mathcal{F}^v_i = \left(x_i, y_i, \mathcal{M}(I_i), \mathcal{F}^v_{mfsd}\left(q_i\right), Z^{loc}_i, I^{loc}_i, D^{loc}_i\right) \tag{3}$$

where $\mathcal{F}^v_{mfsd}\left(q_i\right)$, Z^{loc}_i, I^{loc}_i, and D^{loc}_i calculate the number of points, elevation context features, intensity context features, and local distribution features in the neighborhood of q_i, respectively. Thus, in the proposed SCL-GCN, the channel of the vanilla feature is 16, $\mathcal{F}^v \in \mathbb{R}^{N\times 16}$.

Stratified features. For the vanilla features, each point focuses only on a fixed-size neighborhood, and no relative position can be used. As a result, models trained with vanilla features fail to capture multi-resolution contextual correlation and relative point locations, resulting in models that suffer from a limited effective receptive field, leading to incorrect predictions with complex cracks.

However, our previous study (Feng et al., 2022) showed that simply increasing the neighborhood size of each point in the downsampled pavement point cloud did not improve the model performance. In addition, using the raw (unsampled) pavement point cloud as input while enriching the local feature representation of each point dramatically increases the storage and computational costs. Inspired by Lai et al. (2022), we used a stratified strategy for multi-scale neighborhood sampling from the downsampled road pavement $\mathcal{P}$ and the raw road pavement $\mathcal{P}_o$ to efficiently collect a multi-resolution background. The proposed stratify strategy is shown in Fig. 2.

First, for each point $q_i \in \mathcal{Q}$, its sparse neighborhood $\mathcal{K}^{sparse}_i$ is found in $\mathcal{Q}$ according to the size of the sparse cube S^{sparse}_{cube}. The corresponding dense neighborhood S^{dense}_{cube} is then found in $\mathcal{Q}_o$ using a smaller dense cube size S^{dense}_{cube}.

Next, the relative positions of q_i in the sparse and dense cubes are calculated as follows:

$$P^{sparse}_i = q_i \ominus \mathcal{K}^{sparse}_i \tag{4}$$

$$P^{dense}_i = q_i \ominus \mathcal{K}^{dense}_i \tag{5}$$

where $\ominus$ calculates the element-wise subtraction.

The stratified features $\mathcal{F}^s_i \in \mathbb{R}^{1\times D_r}$ of q_i are obtained by concatenating the vanilla features $\mathcal{F}^v_i$ and the relative positions as follows:

$$\mathcal{F}^s_i = \mathcal{F}^v_i \oplus \mathcal{F}^{RP}_i \tag{6}$$

where $\oplus$ indicates the concatenation operation, and $\mathcal{F}^{RP}_i = P^{sparse}_i \oplus P^{dense}_i$. As the $\mathcal{F}^{RP}_i$ consists of two three-dimensional coordinates, $D_r = 6$. Thus, the channels of the stratified features in this study were 22 and $\mathcal{F}^s \in \mathbb{R}^{N\times 22}$.

Using the stratification strategy with multi-scale neighborhood sampling, the stratified features can effectively aggregate multi-resolution contexts based on relative location information, significantly expanding the effective receptive field of the model.

3.3. Graph feature contrastive learning

The proposed SCL-GCN contained two GCN branches and a contrastive learning mechanism. The two branches had the same architecture, and were trained with different inputs. A contrastive learning mechanism was used to supervise the training processes of the two branches.

In this study, we used ChebyNet (Defferrard et al., 2016) as the basic network of the two GCN branches. The two branches input the same adjacency matrix $\mathcal{A} \in \mathbb{R}^{N\times N}$, but differed in their features. The first branch that input $\mathcal{F}^s$ was denoted as a stratified branch; the second branch that input $\mathcal{F}^v$ was denoted as the vanilla branch. The architecture of the SCL-GCN is shown in Fig. 1.

3.3.1. Dual-branch graph convolution networks

In the GCN, the generalized formulation of the convolution result of the l_{th} layer is $Z^{(l+1)}$, and can be written as:

$$Z^{(l+1)} = \widetilde{D}^{-\frac{1}{2}}\widetilde{\mathcal{A}}\widetilde{D}^{-\frac{1}{2}}Z^{(l)}\Theta^{(l)} \tag{7}$$

where $\Theta^{(l)}$ is the filter parameter matrix of the l_{th} layer. According to Eq. (2), $\mathcal{A}$ is a symmetric matrix. The degree matrix of $\mathcal{A}$ is $\mathcal{D}$, with $\mathcal{D}_{ii} = \sum_j \mathcal{A}_{ij}$. $\mathcal{I}_N$ denotes the identity matrix, define $\widetilde{\mathcal{A}} = \mathcal{A} + \mathcal{I}_N$. The degree matrix of $\widetilde{\mathcal{A}}$ is $\widetilde{\mathcal{D}}$, with $\widetilde{\mathcal{D}}_{ii} = \sum_j \widetilde{\mathcal{A}}_{ij}$.

Denoting $Z^{(L)}_v$ and $Z^{(L)}_s$ as the final convolution results of the vanilla and stratified branch, we obtain

$$Z^{(L)}_* = \left(\widetilde{D}^{-\frac{1}{2}}\widetilde{\mathcal{A}}\widetilde{D}^{-\frac{1}{2}}\right)^{(L-1)}\mathcal{F}^*\Theta \tag{8}$$

where $*= v$ in the vanilla branch, and $*= s$ in the stratified branch. The final filter parameter matrix Θ is the product of the filter parameter matrices of all previous layers and is defined as $\Theta \triangleq \Theta^{L-1}\Theta^{L-2}\cdots\Theta^0$.

3.3.2. Contrastive learning

Vanilla and stratified features have different emphases during the GCN training. Vanilla features provide descriptive local contextual features, whereas stratified features supply multi-scale and long-range relative positions. We applied a contrastive learning mechanism to supervise the dual-branch training process to take full advantage of the vanilla and stratified features. The contrastive learning mechanism calculates the contrastive loss between the vanilla and stratified branches, and rewards or penalizes the vanilla branch learnable loss accordingly. This aims to cluster similar features learned in both branches together in the feature space, while keeping the different features separated.

We denote the loss function of the vanilla and stratified branches as $\mathcal{L}_v$ and $\mathcal{L}_s$, respectively. Accordingly, the corresponding loss can be calculated as:

$$\mathcal{L}_* = -\sum_{m=1}^{C} Y_m \ln\left(softmax\left(Z_*^{(L)}\right)_m\right) \tag{9}$$

where C is the number of categories($C = 2$), in the pavement crack-detection used in this study. $Y = \{Y_1, Y_2\}^T$ is the ground truth in a one-hot form. Thus, the loss function can be further written as:

$$\mathcal{L}_* = -Y_1 \ln\left(softmax\left(Z_*^{(L)}\right)_1\right) - Y_2 \ln\left(softmax\left(Z_*^{(L)}\right)_2\right) \tag{10}$$

where $\left(Z_*^{(L)}\right)_1$ is the convolution result of the "crack" category and $\left(Z_*^{(L)}\right)_2$ is the convolution result of the "non-crack" category. The initial output $Z^{(0)} = \mathcal{F}^*$, where $*= v$ in the vanilla branch, and $*= s$ in the stratified branch.

The contrastive loss is denoted as $\mathcal{L}_{L1}$, the learnable loss of the vanilla branch is rewarded or penalized accordingly, calculated as:

$$\mathcal{L}_{L1} = \left\|\mathcal{L}_v, \mathcal{L}_s\right\|_{L1} \tag{11}$$

where $\left\|\mathcal{L}_v, \mathcal{L}_s\right\|_{L1}$ calculate the least absolute error between $\mathcal{L}_v$ and $\mathcal{L}_s$. Thus, the final loss of the proposed SCL-GCN is denoted as $\mathcal{L}_c$, where $\mathcal{L}_c = \mathcal{L}_s + \mathcal{L}_{L1}$.

4. Experiments

This section presents the experimental data, evaluation metrics, and implementation details and further assesses and analyzes the performance of the SCL-GCN. Sections Section 4.1, 4.2, and 4.3 describe the experimental data, evaluation metrics, and implementation details, respectively. Section 4.4 determines the optimal parameter combination. Section 4.5 evaluates the performance of the SCL-GCN using point clouds. Section 4.6 presents a comparative study of the developed models. Ablation experiments are performed in Section 4.7 to validate the efficacy of the SCL-GCN components. Section 4.8 analyzes the computational efficiency.

4.1. Experimental data

The experimental data for this study were selected from the pavement point cloud dataset (Feng et al., 2022), which was collected from the Qinghai-Tibet highway using a mapping-grade MLS system in September 2015. As the longest (1937 km) and highest (5232 m) asphalt road in the world, the different sections of the Qinghai-Tibet highway are subject to a variety of severe weather, freeze-thaw, and terrain changes. Thus, regular, timely, and comprehensive maintenance of the Qinghai-Tibet highway is far more challenging than for roads in urban and plain areas.

The mapping-grade MLS system, RIEGL VMX-450 was used for data acquisition, consisting of two high-end VQ-450 scanners, four high-resolution cameras, and an inertial navigation unit. During data collection, the RIEGL VMX-450 was mounted on top of a Buick MPV. With a maximum scanning speed of 400 lines/second and an effective range of 800 m, the RIEGL VMX-450 can deliver 1,100,000 measurements/second. The average driving speed during data collection was approximately 80 km/h, ensuring millimeter-level resolution of the collected pavement point clouds.

Table 1
Parameter setting of SCL-GCN.

Parameters	Value
Learning rate	0.01
Units in layer l	2
Dropout rate	0.5
Branch loss	$L2$ loss
Weight for $L2$ loss	5×10^{-4}
Chebyshev polynomial degree	3
Contrastive loss	$L1$ loss
Minimum downsampling distance sd	0.05 m
Radius scale T	3
Fr-NN radius parameter t	2
Sparse cube size S_{cube}^{sparse}	0.3 m
Dense cube size S_{cube}^{dense}	0.1 m

The experimental data comprised 115 pavement point cloud segments with an average of 200,000 points, measuring 8 meters in width and 10 meters in length. In addition, each segment was manually annotated as "crack" or "non-crack" point by point, with an approximate ratio of 2:8. The experimental data were split into Dataset-I (80 pieces) and Dataset-II (35 pieces). Dataset-II was divided into part1, part2, and part3 subsets, according to the increasing order of X and Y coordinates. Part1 contained the first 10% of points, part2 contained the last 60%, and part3 contained the remaining points.

During data pre-processing, we downsampled each pavement segment with the minimum downsampling distance sd to obtain an aligned point cloud, for two main reasons: (1) the "X-type" scanning mode of the two VQ-450s makes the initially acquired pavement point clouds appear in a grid-like arrangement, and (2) the GCN has a strict limitation on the maximum number of input points. Thus, we used downsampling to facilitate subsequent operations, decreasing the computational complexity of the proposed model.

4.2. Evaluation metrics

In this study, metrics including precision (P_{re}), recall (R_{ec}), and F_1-$score$ were used to quantitatively evaluate the performance of the proposed SCL-GCN. These metrics are expressed as follows:

$$P_{re} = \frac{TP}{TP + FP} \tag{12}$$

$$R_{ec} = \frac{TP}{TP + FN} \tag{13}$$

$$F_1\text{-}score = \frac{2 \times Pre \times Rec}{Pre + Rec} \tag{14}$$

where TP, FP, and FN represent the corresponding detected point numbers of the true-positive, false-positive, and false-negative points, respectively. P_{re} calculates the percentage of properly predicted pavement cracks to assess the validity of the model. R_{ec} measures the number of correct positive crack points among all crack points to assess the completeness of the crack detection results. The F_1-$score$ is a function of P_{re} and R_{ec} and measures the overall performance.

4.3. Implementation details

The implementation environment of SCL-GCN is based on Python 3.8.8, Pytorch 1.10.0, Ubuntu 20.04.1, Intel (R) Core (TM) i7-10700K 8-core CPU @ 3.80 GHz, Nvidia RTX 3090 and 64 GB RAM. In addition, each branch of SCL-GCN consists of a 2-layer ChebyNet. The hyperparameters involved in SCL-GCN are optimized in Section 4.4 and listed in Table 1. The proposed SCL-GCN model was trained with part1 of Dataset-II, tested with part2 of Dataset-II, and validated with part3 of Dataset-II.

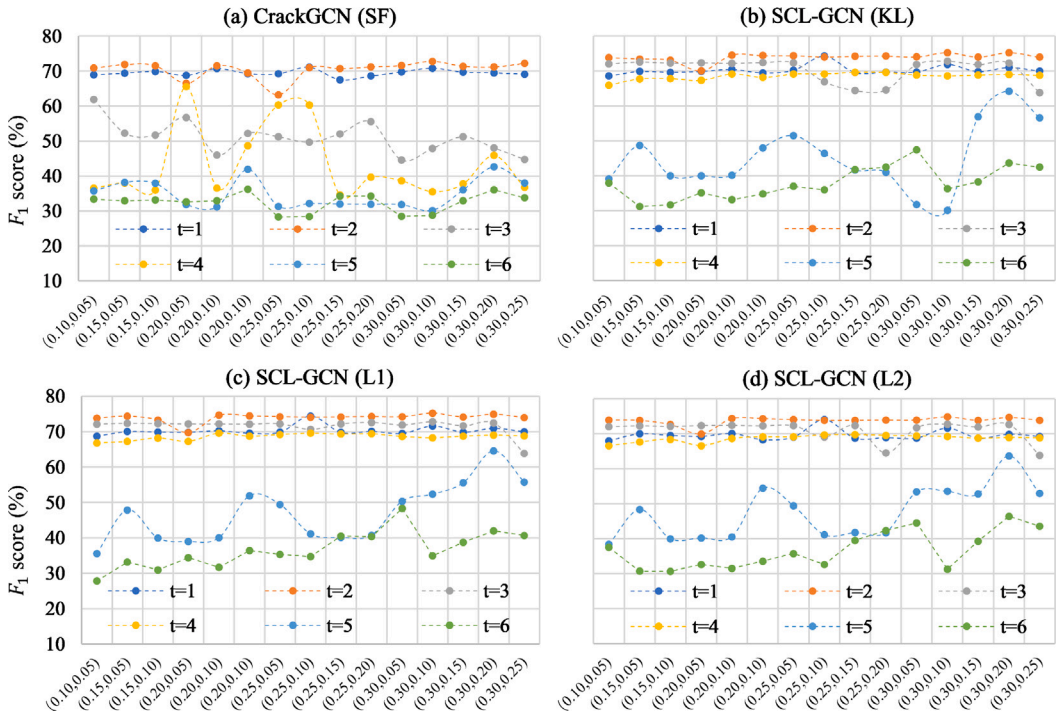


Fig. 3. F_1-*score* (%) of models with different t values and cube size combinations (S_{cube}^{sparse}, S_{cube}^{dense}) (m).

Table 2
F_1-*score* (%) for models with different loss functions and stratified cube sizes when $t = 2$.

(S_{win}, S_{win}^{small}) (m)	CrackGCN (SF)	SCL-GCN (KL)	SCL-GCN (L1)	SCL-GCN (L2)
(0.10, 0.05)	70.8	73.8	73.9	**74.0**
(0.15, 0.05)	71.9	73.5	**74.4**	73.9
(0.15, 0.10)	71.5	73.1	**73.3**	72.7
(0.20, 0.05)	66.5	70.0	69.7	**70.0**
(0.20, 0.10)	71.5	74.6	**74.7**	74.4
(0.20, 0.15)	69.5	74.4	**74.5**	74.4
(0.25, 0.05)	63.2	**74.4**	74.2	74.1
(0.25, 0.10)	71.0	74.0	**74.1**	73.9
(0.25, 0.15)	70.7	74.2	**74.2**	73.9
(0.25, 0.20)	71.2	74.3	**74.3**	74.0
(0.30, 0.05)	71.6	74.1	**74.2**	74.0
(0.30, 0.10)	72.8	**75.2**	**75.2**	74.9
(0.30, 0.15)	71.4	74.0	**74.1**	73.9
(0.30, 0.20)	71.2	**75.2**	74.9	74.7
(0.30, 0.25)	72.2	**74.0**	73.9	73.9
Avg.	70.5	73.9	**74.0**	73.8

4.4. Parameter optimization

SCL-GCN has several important parameters: the sparse cube size S_{cube}^{sparse}, dense cube size S_{cube}^{dense}, and fr-NN search radius R. They have varying effects on the model, S_{cube}^{sparse} and S_{cube}^{dense} determine the descriptiveness of the multi-scale features constructed by the stratification strategy, whereas R affects the receptive field of the convolutions. Generally, the larger the S_{cube}^{sparse} and S_{cube}^{dense}, the more comprehensive the point distribution information represented in feature construction. A larger R results in a wider convolutional receptive field; the loss function used for the contrastive learning module affects the descriptive ability of the feature.

We investigated the influence of these parameters on experimental data to determine the optimal combination of parameters. CrackGCN (SF) represents CrackGCN with stratified features as inputs, whereas SCL-GCN (KL), SCL-GCN (L1), and SCL-GCN (L2) represent models using Kullback–Leibler divergence (KL), mean absolute error (L1), and mean squared error (L2) as contrastive loss functions, respectively.

Fr-NN radius R. The fr-NN search radius $R = t \times sd \times \sqrt{3} \times T$, where $sd = 0.05$ m is the minimum downsampling distance, and $T = 3$ is the optimal fr-NN search radius scale for CrackGCN. Thus, R varied with t. We set the ranges of t to [1, 2, 3, 4, 5, 6] to test the effect of different radii. Table 3 presents the quantity results for the different models with different t and cube-size combinations. Fig. 3 shows the corresponding visualization results, with subplots representing the performance of the models with different contrastive loss functions. With the same S_{cube}^{sparse} and S_{cube}^{dense}, the F_1-*scores* of the different models are different, indicating that the value of t influences the performance of the SCL-GCN. From the first row of each subplot in Fig. 3, it is observed that the best results occur when $t = 2$.

Contrastive loss function. We set $t = 2$ to fix the fr-NN radius, and further explored the influence of different contrastive loss functions with different cube-size combinations. As shown in Table 2, when $t = 2$, SCL-CGN (L1) achieves the 10 best results of the 15 combinations of S_{cube}^{sparse} and S_{cube}^{dense}, and obtain the best average F_1-*score* compared to other models.

Sampling cube sizes. Stratified cubes can provide multi-scale features that enlarge the respective network fields. However, determining the appropriate cubic size requires considerable time and effort. A cubic size that is too small cannot provide sufficient neighboring points to

Table 3
F_1-*score* (%) of models with different t and cube size combinations (S_{cube}^{sparse}, S_{cube}^{dense}) (m).

crackGCN (SF)															
t	(0.10, 0.05)	(0.15, 0.05)	(0.15, 0.10)	(0.20, 0.05)	(0.20, 0.10)	(0.20, 0.15)	(0.25, 0.05)	(0.25, 0.10)	(0.25, 0.15)	(0.25, 0.20)	(0.30, 0.05)	(0.30, 0.10)	(0.30, 0.15)	(0.30, 0.20)	(0.30, 0.25)
1	69.0	69.4	69.9	**68.8**	70.7	69.2	**69.2**	**71.1**	67.5	68.6	69.7	70.8	69.7	69.5	69.1
2	**70.8**	**71.9**	**71.5**	66.5	**71.5**	**69.5**	63.2	71.0	**70.7**	**71.2**	**71.6**	**72.8**	**71.4**	**71.2**	**72.2**
3	61.9	52.3	51.7	56.7	46.0	52.2	51.2	49.7	52.1	55.6	44.6	47.9	51.2	48.0	44.7
4	36.5	38.0	36.0	65.5	36.5	48.7	60.3	60.3	34.6	39.7	38.6	35.5	37.7	45.9	36.7
5	35.8	38.2	37.9	31.9	31.1	41.9	31.3	32.1	32.0	31.9	31.8	30.1	36.0	42.7	38.0
6	33.4	32.9	33.1	32.6	32.9	36.2	28.3	28.4	34.3	34.2	28.4	28.7	33.0	36.0	33.8
SCL-GCN (KL)															
t	(0.10, 0.05)	(0.15, 0.05)	(0.15, 0.10)	(0.20, 0.05)	(0.20, 0.10)	(0.20, 0.15)	(0.25, 0.05)	(0.25, 0.10)	(0.25, 0.15)	(0.25, 0.20)	(0.30, 0.05)	(0.30, 0.10)	(0.30, 0.15)	(0.30, 0.20)	(0.30, 0.25)
1	68.6	69.9	69.6	69.9	70.5	69.5	70.2	74.3	69.6	69.7	69.7	71.9	69.9	71.0	70.0
2	**73.8**	**73.5**	**73.1**	70.0	**74.6**	**74.4**	**74.4**	**74.0**	**74.2**	**74.3**	**74.1**	**75.2**	**74.0**	**75.2**	**74.0**
3	72.0	72.6	72.2	**72.3**	72.2	72.4	72.3	66.9	64.4	64.6	71.9	72.8	71.9	72.3	63.8
4	65.9	67.7	67.8	67.3	69.2	68.2	69.1	69.2	69.6	69.6	68.8	68.6	68.8	69.0	68.8
5	39.1	48.7	40.0	40.0	40.2	48.0	51.5	46.4	41.7	41.0	31.8	30.1	56.9	64.2	56.6
6	37.9	31.2	31.7	35.2	33.2	34.9	37.0	36.0	41.8	42.5	47.5	36.3	38.3	43.7	42.5
SCL-GCN (L1)															
t	(0.10, 0.05)	(0.15, 0.05)	(0.15, 0.10)	(0.20, 0.05)	(0.20, 0.10)	(0.20, 0.15)	(0.25, 0.05)	(0.25, 0.10)	(0.25, 0.15)	(0.25, 0.20)	(0.30, 0.05)	(0.30, 0.10)	(0.30, 0.15)	(0.30, 0.20)	(0.30, 0.25)
1	68.7	70.0	69.9	69.8	70.2	69.6	69.9	**74.4**	69.8	70.1	69.5	71.6	69.9	70.9	70.0
2	**73.9**	**74.4**	**73.3**	69.7	**74.7**	**74.5**	**74.2**	74.1	**74.2**	**74.3**	**74.2**	**75.2**	**74.1**	**74.9**	**73.9**
3	72.1	72.4	72.3	**72.2**	72.2	72.1	72.2	70.6	72.2	72.6	71.9	72.8	71.7	72.5	63.8
4	66.8	67.2	68.2	67.3	69.6	68.7	69.2	69.6	69.3	69.4	68.6	68.3	68.7	69.0	68.8
5	35.6	47.9	40.0	39.0	40.1	51.9	49.4	41.1	40.1	40.8	50.4	52.3	55.6	64.6	55.7
6	27.8	33.2	31.0	34.4	31.7	36.4	35.3	34.8	40.5	40.3	48.3	35.0	38.7	42.0	40.7
SCL-GCN (L2)															
t	(0.10, 0.05)	(0.15, 0.05)	(0.15, 0.10)	(0.20, 0.05)	(0.20, 0.10)	(0.20, 0.15)	(0.25, 0.05)	(0.25, 0.10)	(0.25, 0.15)	(0.25, 0.20)	(0.30, 0.05)	(0.30, 0.10)	(0.30, 0.15)	(0.30, 0.20)	(0.30, 0.25)
1	68.0	70.0	69.5	69.3	70.1	68.4	69.1	**74.1**	68.8	68.9	68.8	71.6	68.8	69.9	69.2
2	**74.0**	**73.9**	**72.7**	70.0	**74.4**	**74.4**	**74.1**	73.9	**73.9**	**74.0**	**74.0**	**74.9**	**73.9**	**74.7**	**73.9**
3	72.1	72.3	72.2	**72.4**	72.4	72.3	72.4	69.1	72.4	64.5	71.8	72.8	72.0	72.7	63.8
4	66.6	67.7	68.4	66.6	68.7	69.1	69.2	69.8	69.8	69.6	69.4	69.2	68.8	69.0	68.9
5	38.4	48.4	40.0	40.2	40.5	54.5	49.4	41.1	41.7	41.7	53.4	53.6	52.8	63.7	52.9
6	37.5	30.7	30.6	32.6	31.5	33.6	35.7	32.6	39.5	42.3	44.5	31.3	39.3	46.4	43.6

capture local spatial geometric information. In contrast, a large cubic size results in redundant or invalid information, degrading accuracy and efficiency. We set the range of the sparse cube sizes S_{cube}^{sparse} to [0.10, 0.15, 0.20, 0.25, 0.30] and the dense cube sizes S_{cube}^{dense} to [0.05, 0.10, 0.15, 0.20, 0.25] in meters. The combination of S_{cube}^{sparse} and S_{cube}^{dense} is denoted as (S_{cube}^{sparse}, S_{cube}^{dense}). We fixed the value of $t = 2$ to explore the effect of different cube size combinations of the SCL-GCN (L1). Fig. 4 shows the F_1-*score* of the SCL-GCN (L1) with $t = 2$ to fix the radius and different cube sizes. The best result occurs when $S_{cube}^{sparse} = 0.30$ m and $S_{cube}^{dense} = 0.10$ m.

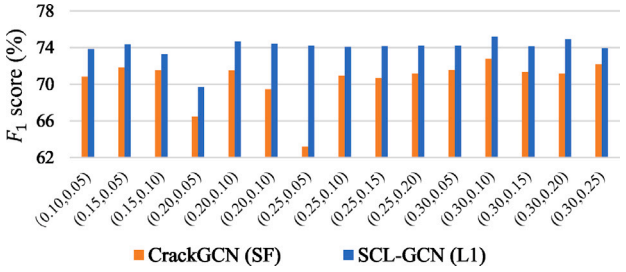


Fig. 4. F_1-*score* of CrackGCN (SF) and SCL-CGN (L1), with $t = 2$ and different cube sizes (S_{cube}^{sparse}, S_{cube}^{dense}).

4.5. Crack detection results

Based on the results of the parametric experiments, we defined the optimal setting for the training process as $t = 2$, L1 loss as the contrastive loss function, the sparse cube size as $S_{cube}^{sparse} = 0.30$ m and the dense cube size as $S_{cube}^{dense} = 0.10$ m. The visual comparison results and corresponding detailed observations of pavement crack detection from MLS point clouds are shown in Fig. 5, which indicates that the developed SCL-GCN model can provide satisfactory crack detection in pavements with complicated cracks. The observations reveal that SCL-GCN achieves outstanding detection results. The experimental results also show that SCL-GCN can effectively distinguish road cracks with varying shapes and sizes. Thus, the SCL-GCN has robust local context, long-range feature construction, and dual-branch contrastive learning capabilities, all of which contribute to accurate and complete point-level crack detection results.

XXX

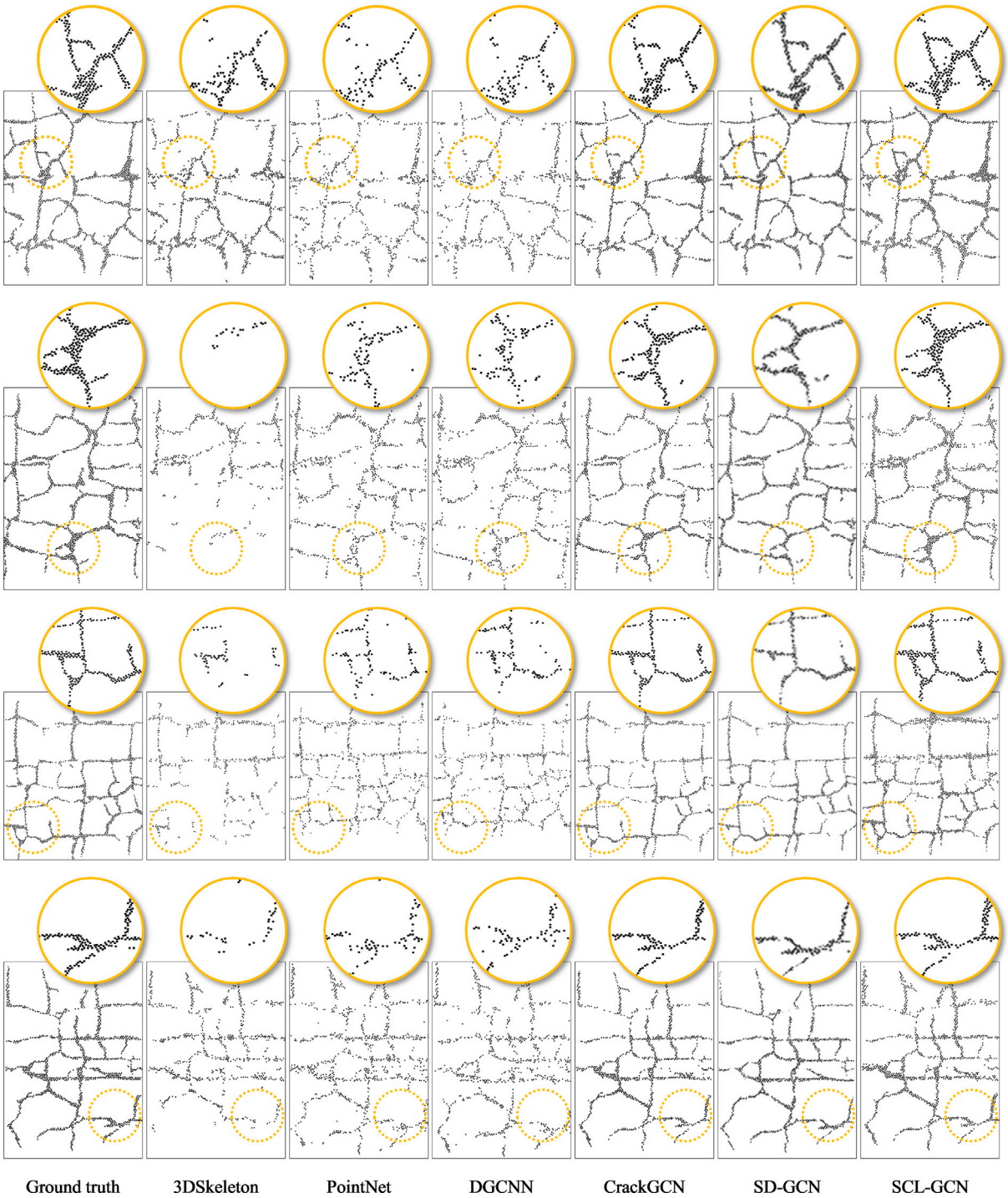


Fig. 5. Visual comparison results and corresponding detailed observations of pavement crack detection using different methods.

4.6. Comparison study

To evaluate the feature descriptive ability of the SCL-GCN in actual environments, we conducted experiments on the pavement point cloud dataset to compare the SCL-GCN with several existing methods. The compared methods included pixel-level methods: U-Net (Ronneberger et al., 2015) and AU-Net (Oktay et al., 2018); a traditional point-level algorithm: 3DSkeleton (Yu et al., 2014); supervised deep learning models: PointNet (Qi et al., 2017) and DGCNN (Wang et al., 2019), and semi-supervised models: CrackGCN (Feng et al., 2022) and SD-GCN (Ma and Li, 2022). For all comparison approaches, we used the same experimental protocols used in CrackGCN and set hyperparameters to their default settings. The experimental results were evaluated based on P_{re}, R_{ec}, and F_1-$score$.

Pixel-level methods. We converted the point cloud pavements into images based on the intensity for training and testing the U-Net and AU-Net models. Pixel-level average evaluation metrics were derived accordingly. However, information loss usually occurs in such approaches due to data dimensionality reduction.

Table 4
Comparison results of different methods for pavement crack detection.

Type	Method	train:test	P_{re}(%)	R_{ec}(%)	F_1-$score$(%)
Pixel	U-Net	7:3	**80.9**	73.3	**76.6**
	AU-Net	7:3	76.5	**77.8**	76.4
Point	3DSkeleton	–	33.6	71.7	46.7
	PointNet	7:3	70.4	63.1	66.1
	DGCNN	7:3	73.0	67.0	70.1
	SD-GCN	7:2	**79.5**	**77.1**	**78.3**
	CrackGCN	**1:6**	70.0	73.9	71.9
	SCL-GCN	**1:6**	75.7	**75.1**	**75.2**

Point-level methods. 3D-Skeleton is a traditional method that combines Otsu thresholding with Euclidean distance-based clustering. We set the density threshold $ds = 1.2$ and the local radius $rd = 0.2$ m for crack detection and outlier removal. However, determining the optimal thresholds in different test scenes is challenging. Although PointNet is a pioneer of point-based deep learning networks, it cannot learn local features efficiently, leading to challenges in fine-grained detection in complex road scenes. Inspired by PointNet and graph neural networks, DGCNN further exploits local geometric structures by introducing dynamic graph edge convolution. Nevertheless, it fails to capitalize on the long-term contextual relationships between intra-class points.

Semi-supervised methods. CrackGCN creates a semi-supervised deep learning framework for detecting pavement cracks based on a graph-widened strategy, whereas SD-GCN further explores a saliency-based dilated graph convolutional network. Due to the finely designed defining features based on graph representation, they achieve more competitive performance than supervised approaches in terms of data dependency and efficiency; however, they fail to account for multi-scale and long-term geometric contexts in pavement point clouds.

In this study, the pixel-level methods were trained on the corresponding intensity images of Dataset-I, the point-level methods were trained on Dataset-I, CrackGCN and the proposed SCL-GCN were trained on the part1 of Dataset-II, and all these methods were tested on the part2 of Dataset-II. However, SD-GCN was trained on the first 70% of Dataset-II and tested on the last 20% of Dataset-II.

Table 4 displays the comparison results based on the average P_{re}, R_{ec}, and F_1-$score$. For the pixel-level models, U-Net and AU-Net had the same training/testing ratio (7:3), similar training time (> 6 h), and similar average F_1-$score$ of 76.6% and 76.4%, respectively. However, taking advantage of the attention mechanism, AU-Net achieves a 4.35% improvement in the average recall compared to U-Net. As a traditional threshold-based method, 3DSkeleton has an average R_{ec} of 71.7% but an average P_{re} of 33.6%, which indicates that 3DSkeleton misclassifies many pavement points as crack points. For supervised algorithms, the graph-related DGCNN outperformed PointNet in all assessment criteria, implying that graph representations may outperform point-based representations in pavement crack-detection tasks. As a semi-supervised approach, CrackGCN has a training/testing split ratio of 1:6 compared with the 7:3 ratio of PointNet and DGCNN, but with a 5.9% higher average recall and 1.8% higher average F_1-$score$ than DGCNN. Although SD-GCN is also trained semi-supervised and achieves excellent performance, it has a training/testing split ratio of 7:2, which means that it has seven times more training data than CrackGCN but only one-third of the test data. Thus, the SD-GCN is a data-driven approach that does not meet the low data dependency requirements for pavement crack detection in MLS point clouds. With the same data split setting and competitive efficiency as CrackGCN, SCL-GCN achieves much better performance. Specifically, the proposed SCL-GCN achieved 75.7%, 75.1%, and 75.2% for average P_{re}, R_{ec}, and F_1-$score$, 5.7%, 1.2%, and 3.3% higher than those of CrackGCN. Thus, the experimental results demonstrate that SCL-GCN outperforms the comparison methods when considering accuracy, efficiency, and data dependency owing to the enhanced descriptive feature encoding and inference capabilities of the constructed stratified features and multi-scale contrastive learning.

The corresponding visualization results are shown in Fig. 5. The cracks detected by the comparative methods are incomplete, whereas the SCL-GCN network yields more complete cracks with fewer outliers. Overall, the SCL-GCN proved to be a satisfactory solution for detecting cracks from large-scale and disordered MLS pavement point clouds.

4.7. Ablation study

We conducted several ablation experiments to explore the effectiveness of the SCL-GCN component. Expressly, CrackGCN and CrackGCN (SF) indicate that CrackGCN is trained with vanilla-version features and stratified features, respectively; SCL-GCN is the model proposed in this paper. During the experiments, we fixed the same training/testing split ratio of 1:6 and other hyperparameter settings as described in Section 4.4. As shown in Table 5, the crack-detection performance increased gradually with the addition of stratified features and the contrastive learning module. Compared with CrackGCN, CrackGCN (SF) improved the average P_{re} by 6.4% and average F_1-$score$ by approximately 1% at the cost of a 4.3% decrease in the average R_{ec}. The results reveal that the designed stratified features provide a more accurate and specific description of pavement cracks but lead to a decrease in the detection rate for cracks with inconspicuous geometric features. With the addition of the contrastive learning (CL) mechanism, SCL-GCN outperforms CrackGCN by 5.7%, 1.2%, and 3.3% in P_{re}, R_{ec}, and F_1-$score$, respectively, and outperformed CrackGCN (SF) by 5.5% and 2.4% in R_{ec} and F_1-$score$. he proposed contrastive learning module narrows the distance of multi-scale features in the feature space, allowing mutual assistance to improve the overall structural description ability while ensuring the geometric details of pavement cracks. Thus, stratified feature construction and contrastive learning modules are effective.

4.8. Efficiency analysis

Table 6 presents the training times for the different models. U-Net and AU-Net required the longest training times as pixel-level models, exceeding 6 h. Owing to the high computational cost of dynamic graph convolution, the training time of DGCNN was twice as long as that of PointNet, approximately 4 h. In general, the efficiency of the semi-supervised models was significantly higher than that of the supervised models. Among them, CrackGCN was the most efficient. The training time of CrackGCN (SF) was slightly longer than that of CrackGCN due to the additional computations caused by the extra features. The crack detection performance of the proposed SCL-GCN remarkably outperformed that of CrackGCN (SF) due to the contrastive learning mode of the two GCN branches. However, the training time of SCL-GCN was much less than twice that of CrackGCN (SF), only 1.27 times, which proves the compactness and training efficiency of the dual-branch architecture of SCL-GCN. Furthermore, SCL-GCN achieved competitive performance on five times more test data than SD-GCN with a slight increase in training time using only one-seventh of the training data. Thus, the proposed SCL-GCN is an efficient and less-data-dependent solution for pavement crack detection.

Table 5
Ablation experimental results of different methods.

Method	CrackGCN	SF	CL	P_{re}(%)	R_{ec}(%)	F_1-$score$(%)
CrackGCN	✓			70.0	73.9	71.9
CrackGCN (SF)	✓	✓		**76.4**	69.6	72.8
SCL-GCN	✓	✓	✓	75.7	**75.1**	**75.2**

Table 6
The computational costs of different methods measured in training time.

Method	U-Net	AU-Net	PointNet	DGCNN	CrackGCN	CrackGCN(SF)	SD-GCN	SCL-GCN (ours)
Training time	>6 h	>6 h	>2 h	>4 h	~40 min	~45 min	~55 min	~70 min

5. Conclusion

This paper presents SCL-GCN, a novel dual-branch GCN-based architecture consisting of a multi-scale graph representation construction module and a graph feature contrastive learning module for pavement crack detection from MLS point clouds. The multi-scale graph representations construction module enables the model to increase the effective receptive field with long-range contexts at a low computational cost. In addition, the graph feature contrastive learning module supervises the learning process of dual-branch GCNs using stratified features, avoiding learning bias caused by imbalanced data distribution, promoting convergence and improving performance. The experimental results demonstrate that the developed SCL-GCN outperforms state-of-the-art approaches, with comprehensive consideration of accuracy, efficiency, and data dependency. With a training/testing ratio of only 1:6 and an overall training time of less than 70 min, the average precision, recall, and F_1-$score$ of the SCL-GCN reached 75.7%, 75.1%, and 75.2%, respectively. The effectiveness of the modules in the SCL-GCN was analyzed through a series of ablation experiments, further verifying the efficacy of the model. We have developed an efficient technique for improving pavement crack-detection performance using MLS point clouds. Future research will introduce knowledge distillation techniques to monitor and guide networks with more compact architectures to increase the integrity and accuracy of detected cracks.

CRediT authorship contribution statement

Huifang Feng: Conceptualization, Methodology, Investigation, Writing – original draft, Writing – review & editing. **Lingfei Ma:** Conceptualization, Investigation, Writing – review & editing, Funding acquisition, Supervision. **Yongtao Yu:** Conceptualization, Investigation, Supervision. **Yiping Chen:** Conceptualization, Investigation, Supervision. **Jonathan Li:** Conceptualization, Investigation, Funding acquisition, Supervision.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by National Natural Science Foundation of China under Grants 41871380, 42101451, and 62076107; Natural Science Foundation of Jiangsu Province under Grant BK20211365; and the Emerging Interdisciplinary Project of Central University of Finance and Economics in China.

References

Cha, Y.J., Choi, W., Büyüköztürk, O., 2017. Deep learning-based crack damage detection using convolutional neural networks. Comput.-Aided Civ. Infrast. Eng. 32 (5), 361–378. http://dx.doi.org/10.1111/mice.12263.

Cubero-Fernandez, A., Rodriguez-Lozano, F., Villatoro, R., Olivares, J., Palomares, J.M., et al., 2017. Efficient pavement crack detection and classification. EURASIP J. Imag. Video Process. 2017 (1), 1–11. http://dx.doi.org/10.1186/s13640-017-0187-0.

Dabous, S.A., Yaghi, S., Alkass, S., Moselhi, O., 2017. Concrete bridge deck condition assessment using IR thermography and ground penetrating radar technologies. Autom. Constr. 81, 340–354. http://dx.doi.org/10.1016/j.autcon.2017.04.006.

Defferrard, M., Bresson, X., Vandergheynst, P., 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (Eds.), Advances in NeurIPS. 29, Curran Associates, Inc.

Fei, Y., Wang, K.C.P., Zhang, A., Chen, C., Li, J.Q., Liu, Y., Yang, G., Li, B., 2020. Pixel-level cracking detection on 3D asphalt pavement images through deep-learning-based CrackNet-V. IEEE Trans. Intell. Transp. Syst. 21 (1), 273–284. http://dx.doi.org/10.1109/TITS.2019.2891167.

Feng, H., Li, W., Luo, Z., Chen, Y., Fatholahi, S.N., Cheng, M., Wang, C., Junior, J.M., Li, J., 2022. GCN-based pavement crack detection using mobile LiDAR point clouds. IEEE Trans. Intell. Transp. Syst. 23 (8), 11052–11061. http://dx.doi.org/10.1109/TITS.2021.3099023.

Gavilán, M., Balcones, D., Marcos, O., Llorca, D.F., Sotelo, M.A., Parra, I., Ocaña, M., Aliseda, P., Yarza, P., Amírola, A., 2011. Adaptive road crack detection system by pavement classification. Sensors 11 (10), 9628–9657. http://dx.doi.org/10.3390/s111009628.

Guan, H., Li, J., Yu, Y., Chapman, M., Wang, H., Wang, C., Zhai, R., 2015. Iterative tensor voting for pavement crack extraction using mobile laser scanning data. IEEE Trans. Geosci. Remote Sens. 53 (3), 1527–1537. http://dx.doi.org/10.1109/TGRS.2014.2344714.

Hong, S., Wiggenhauser, H., Helmerich, R., Dong, B., Dong, P., Xing, F., 2017. Long-term monitoring of reinforcement corrosion in concrete using ground penetrating radar. Corros. Sci. 114, 123–132. http://dx.doi.org/10.1016/j.corsci.2016.11.003.

Hou, Y., Li, Q., Zhang, C., Lu, G., Ye, Z., Chen, Y., Wang, L., Cao, D., 2021. The state-of-the-art review on applications of intrusive sensing, image processing techniques, and machine learning methods in pavement monitoring and analysis. J. Eng. 7 (6), 845–856. http://dx.doi.org/10.1016/j.eng.2020.07.030.

Jiang, H., Li, Q., Jiao, Q., Wang, X., Wu, L., 2018. Extraction of wall cracks on earthquake-damaged buildings based on TLS point clouds. IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens. 11 (9), 3088–3096. http://dx.doi.org/10.1109/JSTARS.2018.2857564.

Kang, D., Benipal, S.S., Gopal, D.L., Cha, Y.-J., 2020. Hybrid pixel-level concrete crack segmentation and quantification across complex backgrounds using deep learning. Autom. Constr. 118, 103291. http://dx.doi.org/10.1016/j.autcon.2020.103291.

Lai, X., Liu, J., Jiang, L., Wang, L., Zhao, H., Liu, S., Qi, X., Jia, J., 2022. Stratified transformer for 3d point cloud segmentation. In: CVPR. pp. 8500–8509.

Lim, R.S., La, H.M., Shan, Z., Sheng, W., 2011. Developing a crack inspection robot for bridge maintenance. In: ICRA. pp. 6288–6293. http://dx.doi.org/10.1109/ICRA.2011.5980131.

Lim, R.S., La, H.M., Sheng, W., 2014. A robotic crack inspection and mapping system for bridge deck maintenance. IEEE Trans. Autom. Sci. Eng. 11 (2), 367–378. http://dx.doi.org/10.1109/TASE.2013.2294687.

Liu, Z., Cao, Y., Wang, Y., Wang, W., 2019. Computer vision-based concrete crack detection using U-net fully convolutional networks. Autom. Constr. 104, 129–139. http://dx.doi.org/10.1016/j.autcon.2019.04.005.

Liu, F., Liu, J., Wang, L., 2022. Asphalt pavement crack detection based on convolutional neural network and infrared thermography. IEEE Trans. Intell. Transp. Syst. 23 (11), 22145–22155. http://dx.doi.org/10.1109/TITS.2022.3142393.

Liu, J., Yang, X., Lau, S., Wang, X., Luo, S., Lee, V.C.-S., Ding, L., 2020. Automated pavement crack detection and segmentation based on two-step convolutional neural network. Comput.-Aided Civ. Infrast. Eng. 35 (11), 1291–1305. http://dx.doi.org/10.1111/mice.12622.

Ma, L., Li, J., 2022. SD-GCN: Saliency-based dilated graph convolution network for pavement crack extraction from 3D point clouds. Int. J. Appl. Earth Obs. Geoinf. 111, 102836. http://dx.doi.org/10.1016/j.jag.2022.102836.

Ohno, K., Ohtsu, M., 2010. Crack classification in concrete based on acoustic emission. Constr. Build. Mater. 24 (12), 2339–2346. http://dx.doi.org/10.1016/j.conbuildmat.2010.05.004.

Oktay, O., Schlemper, J., Folgoc, L.L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N.Y., Kainz, B., Glocker, B., Rueckert, D., 2018. Attention U-Net: Learning where to look for the pancreas. http://dx.doi.org/10.48550/arXiv.1804.03999, arXiv:1804.03999.

Oliveira, H., Correia, P.L., 2009. Automatic road crack segmentation using entropy and image dynamic thresholding. In: EUSIPCO. pp. 622–626.

Peng, L., Chao, W., Shuangmiao, L., Baocai, F., 2015. Research on crack detection method of airport runway based on twice-threshold segmentation. In: IMCCC. IEEE, pp. 1716–1720.

Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017. PointNet: Deep learning on point sets for 3D classification and segmentation. In: CVPR. pp. 652–660.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-Net: Convolutional networks for biomedical image segmentation. In: MICCAI. Springer, pp. 234–241. http://dx.doi.org/10.1007/978-3-319-24574-4_28.

Salman, M., Mathavan, S., Kamal, K., Rahman, M., 2013. Pavement crack detection using the gabor filter. In: In Proc. IEEE ITSC. IEEE, pp. 2039–2044. http://dx.doi.org/10.1109/ITSC.2013.6728529.

Tong, Z., Yuan, D., Gao, J., Wei, Y., Dou, H., 2020. Pavement-distress detection using ground-penetrating radar and network in networks. Constr. Build. Mater. 233, 117352. http://dx.doi.org/10.1016/j.conbuildmat.2019.117352.

Turkan, Y., Hong, J., Laflamme, S., Puri, N., 2018. Adaptive wavelet neural network for terrestrial laser scanner-based crack detection. Autom. Constr. 94, 191–202. http://dx.doi.org/10.1016/j.autcon.2018.06.017.

Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M., 2019. Dynamic graph CNN for learning on point clouds. ACM Trans. Graph. 38 (5), http://dx.doi.org/10.1145/3326362.

Xu, X., Yang, H., 2019. Intelligent crack extraction and analysis for tunnel structures with terrestrial laser scanning measurement. Adv. Mech. Eng. 11 (9), http://dx.doi.org/10.1177/1687814019872650.

Yang, F., Zhang, L., Yu, S., Prokhorov, D., Mei, X., Ling, H., 2020. Feature pyramid and hierarchical boosting network for pavement crack detection. IEEE Trans. Intell. Transp. Syst. 21 (4), 1525–1535. http://dx.doi.org/10.1109/TITS.2019.2910595.

Yu, Y., Li, J., Guan, H., Wang, C., 2014. 3D crack skeleton extraction from mobile LiDAR point clouds. In: IGARSS. pp. 914–917. http://dx.doi.org/10.1109/IGARSS.2014.6946574.

Zhang, A., Wang, K.C.P., Fei, Y., Liu, Y., Tao, S., Chen, C., Li, J.Q., Li, B., 2018. Deep learning-based fully automated pavement crack detection on 3D asphalt surfaces with an improved CrackNet. J. Comput. Civ. Eng. 32 (5), http://dx.doi.org/10.1061/(ASCE)CP.1943-5487.0000775.

Zhang, A., Wang, K.C., Li, B., Yang, E., Dai, X., Peng, Y., Fei, Y., Liu, Y., Li, J.Q., Chen, C., 2017. Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network. Comput.-Aided Civ. Infrast. Eng. 32 (10), 805–819. http://dx.doi.org/10.1111/mice.12297.

Zou, Q., Cao, Y., Li, Q., Mao, Q., Wang, S., 2012. CrackTree: Automatic crack detection from pavement images. Pattern Recognit. Lett. 33 (3), 227–238. http://dx.doi.org/10.1016/j.patrec.2011.11.004.