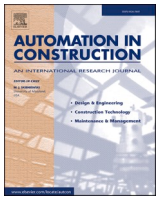
Automation in Construction 159 (2024) 105282

Contents lists available at ScienceDirect

Automation in Construction

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



Dynamic graph CNN based semantic segmentation of concrete defects and as-inspected modeling

Fardin Bahreini [a], Amin Hammad [b,*]

[a] Concordia University, Department of Building, Civil & Environmental Engineering, Canada
[b] Concordia University, Concordia Institute for Information Systems Engineering, Canada

ARTICLE INFO

Keywords:
Concrete surface defect
Semantic segmentation
3D point cloud
DGCNN
As-inspected modeling

ABSTRACT

Obtaining accurate information of defective areas of infrastructures helps to perform repair actions more efficiently. Recently, LiDAR scanners have been used for the inspection of surface defects. Moreover, machine learning methods have attracted the attention of researchers for semantic segmentation and classification based on point cloud data. Although much work has been done for processing visual information with images, research on machine learning methods for semantic segmentation of raw point cloud data is still in its early stages. Moreover, LiDAR technology is commonly used to create as-is BIM models. Therefore, the BIM model needs to be integrated with the results of defect semantic segmentation after the LiDAR-based inspection. Addressing the above issues, this paper has the following objectives: (1) Developing a method for point cloud-based concrete surface defects semantic segmentation; and (2) Developing a semi-automated process for as-inspected modeling. The challenges related to the size of the dataset and imbalanced classes are studied. Sensitivity analysis is applied to capture the best combination of hyperparameters and investigate their effects on the network performance. The proposed method resulted in 98.56% and 96.50% recalls for semantic segmentation of cracks and spalls, respectively. Furthermore, post-processing of the results of the concrete surface defects semantic segmentation is done to semi-automate the process of as-inspected modeling. As-inspected BIM includes the updated information of the facilities at the time of data collection. This semi-automated process made it possible to manage and visualize the detected defects by extracting their dimensions and identifying the conditions on the 3D model.

1. Introduction

Advanced technologies (e.g., scanners, sensors) have made the inspection process more accurate and reliable [1]. Light Detection and Ranging (LiDAR) scanners can collect high-quality 3D point cloud datasets. In order to automate the process of concrete surface inspection, it is important to collect proper datasets and use an efficient approach to analyze them and find the defects. Moreover, Deep Neural Networks (DNNs) have been recently used for detecting 3D objects within point clouds collected by a LiDAR scanner [2]. In order to detect concrete surface defects, the CNN approach can be applied to point cloud datasets. Each neural network is trained for a unique purpose. Adaptation of the right algorithm to a specific purpose can greatly improve the performance. On the other hand, the detected surface defects information can be linked to the BIM model. Therefore, the inspection process can be more efficient by utilizing an integrated process of surface defect semantic segmentation and defect modeling.

The key problems of this research can be attributed to the issues as follows: The first issue is related to the automated concrete surface semantic segmentation of point cloud data. Visual inspection using non-equipped eyes is the principal method of detecting structural surface defects, which is unsafe, time-consuming, expensive, and subjective to human errors [3]. Using remote sensing, such as cameras and LiDAR scanners, is one solution to overcome these shortcomings. The captured point cloud data from the real environment can assist in detecting the defects and taking further actions. It will also provide a database for other maintenance measures after creating an integrated 3D model for existing structures. Image-based inspection using cameras is based on pattern recognition techniques [4]. Several studies focused on detecting defects automatically (e.g. spalling [5] or cracks [6]) and determining some characteristics such as the width of cracks [7,8]. Several crack detection algorithms have been developed, which can be practically used for real-time crack analysis [9,10], crack classification [11], and automating crack sealing [12,13]. There are several challenges in

* Corresponding author.
E-mail addresses: f_bahrei@encs.concordia.ca (F. Bahreini), amin.hammad@concordia.ca (A. Hammad).

https://doi.org/10.1016/j.autcon.2024.105282
Received 11 August 2022; Received in revised form 25 November 2023; Accepted 5 January 2024
Available online 13 January 2024

supporting concrete inspection using image-based methods. Such methods are mostly defined for simple flat concrete surfaces and may fail in analyzing more complex geometries and materials [12]. Good lighting conditions are one of the main issues that should be considered during implementing these methods [14]. Another shortcoming is the necessity of providing supplementary information, such as camera lens, focal length, or the distance from the camera to the target surface, before analyzing the images [15]. Moreover, in comparison to image data, measuring the defects dimensions such as depth is more accurate and reliable in point cloud-based methods. Although the initial cost of LiDAR scanners is more than cameras, it may be more profitable and economical in the long term.

In order to automate the process of point cloud-based inspection, appropriate datasets and an efficient approach such as defect semantic segmentation are essential. Recently, machine learning methods have attracted the attention of researchers for semantic segmentation and classification based on point clouds. Unlike other methods, such as the Hough Transform (HT) [16] and the Random Sample Consensus (RANSAC) approach [17], machine learning methods are robust and flexible. However, they rely on the point cloud density and size of the dataset. Moreover, training based on large datasets is time-consuming [18] and converting the point cloud into other representations increases the dataset size. Different methods such as classification, part segmentation, and semantic segmentation can be used to process the raw point cloud data [19]. This research focuses on semantic segmentation, which is based on the detailed information of each point. Although much work has been done for processing visual information with images, research on machine learning methods for semantic segmentation of raw point cloud data is still in its early stages [20]. Moreover, no deep learning method is currently available for semantic segmentation of the surface defects based on point clouds without converting the raw data to other representations (e.g. images).

The second issue is related to as-inspected modeling. Most existing structures do not have a 3D model; and even when available, it is not a complete model. LiDAR technology is commonly used to create as-is BIM models. However, the as-is model does not include the inspection results [21]. Therefore, the BIM model needs to be integrated with the results of defect semantic segmentation after the LiDAR-based inspection.

Given the problems explained above, the main objectives of this paper are: (1) Developing a method for point cloud-based concrete surface defects semantic segmentation; and (2) Developing a semi-automated process for as-inspected modeling. This paper is an extension of our previous work [22]. This paper focuses on developing a method for point cloud-based defect semantic segmentation called Normal Vector Enhanced Dynamic Graph Convolutional Neural Network (NVE-DGCNN) to automate the inspection process of concrete surface defects, including cracks and spalls. The paper investigates two main characteristics related to surface defects, including normal vector and depth. The paper starts with an adapted DGCNN [19], and then the main network of this research, which is NVE-DGCNN, is investigated. Furthermore, post-processing of the results of the concrete surface defects semantic segmentation is done to semi-automate the process of as-inspected modeling. The rest of the paper is structured as follows: Section 2 contains the literature review. The methodology is explained in Section 3. Section 4 explains data collection and pre-processing, Sections 5 and 6 show the implementation results and the case study. Finally, the conclusions and future work are presented in Section 7.

2. Literature review

2.1. LiDAR-based defect detection

LiDAR scanning is a non-contact measurement technology that has proven its potential in capturing accurate and instant point cloud data from object surfaces [23,24]. However, the resolution and noise level of point cloud data pose some challenges in detecting small cracks [25]. Therefore, to overcome this limitation, an additional feature, which is the RGB color, is considered in deep learning models [3,20]. Various methods have been applied to the point cloud data to detect surface defects. Geometry analysis and machine learning methods are two main approaches for detecting concrete surface defects.

Gaussian curvature distribution can be used to calculate volume loss [26]. Another method to detect concrete surface defects is the crossing section methods [27]. Laefer et al. [15] used fundamental mathematics to define the smallest width of unit-based masonry cracks, which can be detected with LiDAR scanner by considering the main parameters of depth and orientation of crack, orthogonal offset, and interval scan angle. Anil et al. [28] focused on the performance of LiDAR scanners, by using an automated algorithm on point cloud data from reinforced concrete surfaces and asserted the possibility of detecting 1 *mm* crack based on point cloud data. Xu and Yang [29] used the Gaussian filtering method and image-generated data from the point cloud to detect the cracks of a concrete tunnel structure. Teza et al. [26] proposed an automatic method for the inspection of damaged areas of concrete bridge surfaces using a LiDAR scanner and Gaussian mean curvature computation. Makuch and Gawronek [30] proposed an automatic inspection system for reinforced concrete cooling tower shells using point cloud data and local surface curvature computation. Olsen et al. [27] proposed using cross sectional analysis to detect surface damage based on LiDAR scanner data. Liu et al. [31] utilized the distance and gradient-based method to detect the defective area of bridge surfaces using laser scanner data. An automated classification algorithm for detecting historical building defects is suggested by Armesto-Gonzalez et al. [32]. Valença et al. [33] proposed a method combining image processing and LiDAR scanning technology to automate the process of capturing the geometrical characteristics of cracks on concrete bridges. Kim et al. [34] proposed a technique to indicate the location and measure the quantity of concrete surface spalling defects larger than 3 *mm* using LiDAR scanner data. Truong-Hong et al. [24] presented an approach to detect the bridge cracks using a LiDAR scanner and developed a tool to measure the length and width of cracks based on point cloud data and RGB color produced from an external camera. Tsai and Li [35] assessed the probability of using point cloud data to detect cracks with the dynamic-optimization-based segmentation method and assess the cracks segmentation performance using the linear-buffered Hausdorff scoring method. Cabaleiro et al. [36] developed an automatic cracks detection algorithm using LiDAR data for timber beams inspection to identify the cracks geometrical characteristics. Mizoguchi et al. [37] proposed a customized region-growing algorithm along with an iterative closest point algorithm to detect the surface defects of concrete structures based on LiDAR scanner data. Nasrollahi et al. [38] proposed a method for detecting concrete surface defects based on collecting point cloud data from LiDAR scanners and using a Deep Neural Network (DNN). Guldur et al. [3] proposed a method to detect the defects using point clouds' intensity and RGB values to define a threshold and extract the defect's geometrical features [3]. However, their method is not based on using the point cloud geometrical features in the detection process and is not suitable for complicated structures [39]. Guldur and Hajjar [40] developed damage detection algorithms for automatic surface normal-based defect detection and quantification using LiDAR scanner data.

So far, different deep learning methods have been used to identify concrete surface defects using images, and progress in this area has reached an acceptable level [41]. Image-based methods are usually affected by the consistency and stability of light conditions for the captured images, and these methods are usually suitable only for simple flat surfaces [12,14]. Although the papers discussed in this section have significant value in the field of defect detection, there is no deep learning method for semantic segmentation of concrete surface defects using raw point cloud data.

2.2. Semantic segmentation of point clouds

The use of point cloud-based deep learning methods is a breakthrough in identifying concrete surface defects in 3D points data. Point cloud-based deep learning methods are currently in their early stages, and very little research has been done in this area [41]. Data quality is critical in determining the best fitting function for any neural network. The datasets should reflect the appropriate parameters and provide various cases depending on the requirements. As a result, gathering sufficient datasets is required to obtain an accurate model. A point cloud is a set of data that includes the geometric information of sparse points collected in three dimensions. RGB and density information could also be included in point cloud datasets. Pixel-based, voxel-based, and 3D point-based approaches are three main categories of CNN approaches based on data representation. 3D data is transformed into 2D representation in pixel-based approaches [42]. In voxel-based approaches, the 3D points are used to create voxels [43]. Qi et al. [44] presented 3D point-based approaches to process point cloud data using 3D CNN and utilizing 3D recognition tasks such as object classification, part segmentation, and semantic segmentation. Since pixel and voxel-based methods are more common than point-based methods, 3D point cloud datasets are frequently converted into images or 3D-voxel grids before being used in deep learning. In certain circumstances, the transformation produces enormous data with uncertain invariances. An intriguing characteristic of point clouds lies in their inherent invariance under transformations. This property implies that when point clouds are transformed into alternative representations, the underlying geometrical properties of the points may undergo changes. Furthermore, point cloud data are easier to learn because of their clarity and consistent structure, but meshes are complicated and contain contradictory compositional patterns [45].

2.3. Local feature learning on point sets

The Dynamic Graph CNN (DGCNN) is a recent network proposed by Wang et al. [19]. It is a new point-based CNN suitable for high-level tasks, such as object classification and semantic segmentation. DGCNN can improve capturing local geometric functions as it creates a local neighborhood graph and dynamically updates the graph with the nearest neighbors after each layer of the network. DGCNN rather than operating on individual points, iteratively performs convolution on edges, associating the neighborhood point pairs. The operation layer for edge feature generation in DGCNN is called EdgeConv, which can define the relationships between a point and its neighbors [19].

The segmentation model of DGCNN involves a series of three EdgeConv layers and three fully connected layers. The parameter K in the model is the number of the edge features for each point, which is computed in each EdgeConv layer for the input of n points. Edge feature is the most important feature in concrete surface defect semantic segmentation. Wang et al. [19] stated that the model with their developed DGCNN improved the accuracy for classification task in comparison of PointNet++ for the same ModelNet40 [46] dataset. Furthermore, for the semantic segmentation task, they used Stanford large-scale 3D indoor spaces dataset (S3DIS) [47] and compared their work with PoinNet, for which their work achieved a higher accuracy. In another study, Pierdicca et al. [48] compared the performance of PointNet++ and DGCNN for semantic segmentation of historical architectural elements. They used a publicly available digital cultural heritage dataset with 11 labeled points clouds.

2.4. As-inspected modeling

Several studies explored extending BIM for Inspection Information Modeling. *Davila Delgado* et al. [49] *proposed* an extension to the Industry Foundation Classes (IFC) data model standard for structural monitoring systems. Their extension could model the structural monitoring systems, store and retrieve obtained data, and visualize the BIM model's data. Chen et al. [50] proposed an approach to monitor the state of assets using an embedded sensing system and IFC-based BIM model. Hammad et al. [51] proposed a framework for life-cycle infrastructure information modeling and management. However, they did not discuss the details of the formal definition of this information. Some previous work, such as Mailhot and Busuio [52], focused on the manual recording of the inspection data in a 2D or 3D location-based sketching. Hammad et al. [53] demonstrated the applicability of 4D visualization of bridge lifecycle information based on a standard model. They proposed a framework for a mobile model-based bridge lifecycle management system to link all the information related to the design, construction, inspection, and maintenance to a 4D model of the bridge, combining different scales of space and time. However, their proposed system did not include measured data.

Hamledari et al. [54] proposed a computational solution that uses IFC schema to automatically update as-designed BIM based on construction and facility inspection data. Ma et al. [55] proposed an information model based on the IFC schema for damaged reinforced concrete structures from earthquake events. Based on their study, cracks can be represented as a texture on the element's surface, and structural damages such as breakage can be represented by trimming the building elements. Tanaka et al. [56] proposed an information model based on IFC to support the bridge inspection process. Moreover, a web-based system was developed based on Web Graphics Library (WebGL) to show the inspection information and images of degraded parts. Tanaka et al. [57] continued their work and proposed a system to extract inspection and repair reports. However, their work did not cover the detailed semantic information for inspection and repair processes. Sacks et al. [58] proposed a SeeBridge system for bridge information modeling based on inspection data. Their approach was not independent of the type of structure, and a detailed model for defect data was not covered in their work [59]. Hüthwohl et al. [60] proposed a framework to integrate bridge defect information with BIM. In their approach, some defect characteristics (e.g. type and size) were considered, and texture images were used to represent defects. Hamdan and Scherer [61] presented a framework for representing structural damages in BIM. Their approach was based on the multi-model approach, and in their study, a layered structure was developed to represent and visualize the damage geometry. In another study, Hamdan et al. [62] proposed a framework for semi-automatic generation of damage models and machine-based interpretation of the recorded structural damage data using ontology. Their work was based on a linked model approach, and a separate file was used to represent the damage model. This approach does not have all inspection data in one BIM model. Furthermore, their study relied on images that do not represent the defect's geometry as accurately as point cloud data.

Artus and Koch [63] presented two ways for modeling physical defects using IFC based on surface and void approaches. The texture images on top of the 3D component can represent the defect information in the surface-based approach. In the void-based approach, the defect geometry was subtracted from component geometry. Meshes from the point cloud can represent the spalling defect on the void element. Moreover, their void-based approach still has a problem with cracks geometry as cracks were modeled as an extrusion of triangular profile. In another study, Artus et al. [64] presented a framework that generates spalling defect geometries from photos and saves them into a data model using IFC based on surface and void features. However, their work was based on image data, which does not contain the depth of defects. Isailović et al. [65] proposed a use case for enhancing an IFC-based bridge model using the image-based classification to identify the spalling defect features. Their approach depends on collected images from the inspection, and defect characteristics were directly identified from photos, which is not as accurate as point cloud data.

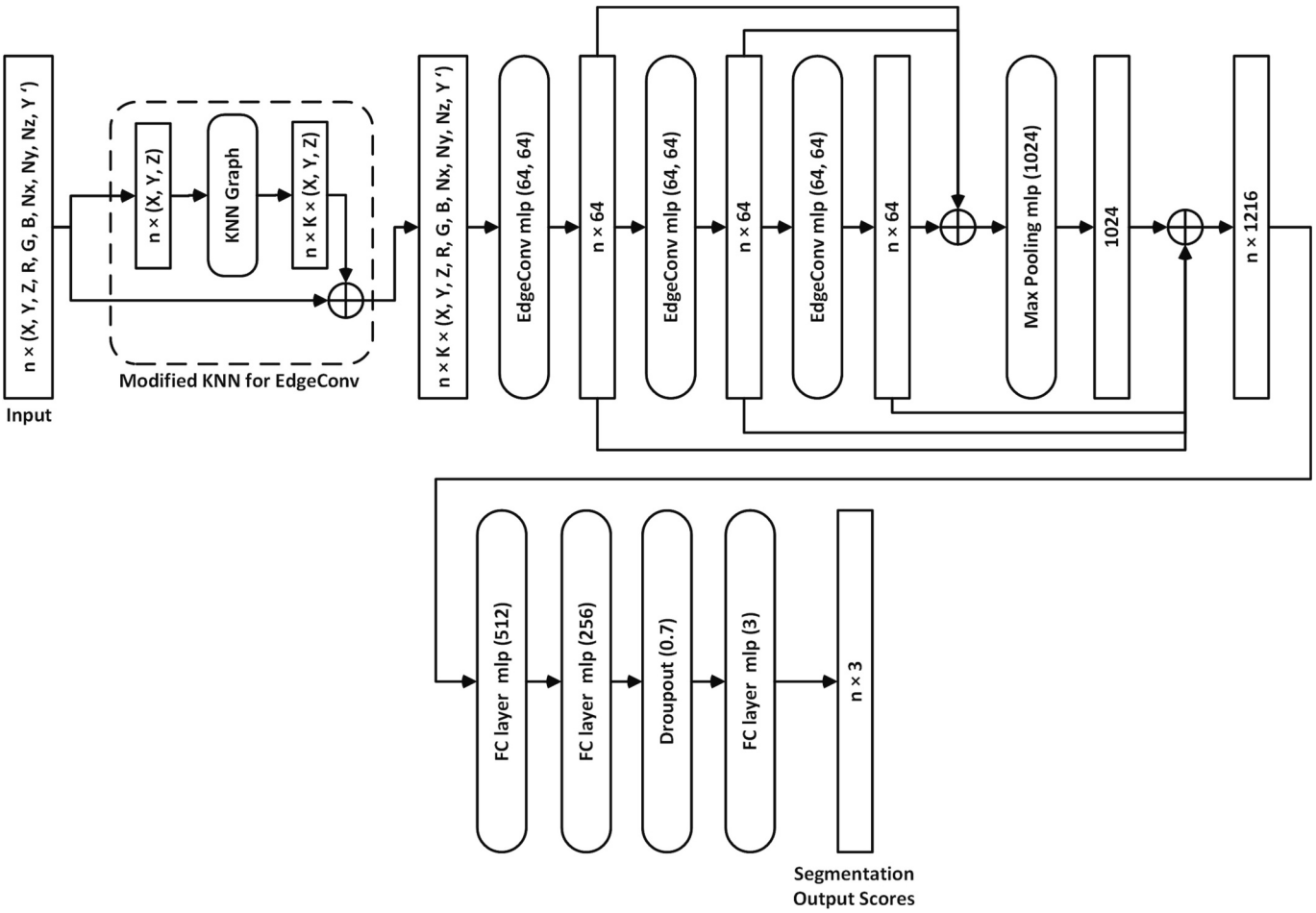


Fig. 1. Architecture of NVE-DGCNN (adapted from [19]).

3. Methodology

As explained in Section 2.3, DGCNN [19] is a deep neural network for classification, part segmentation, and semantic segmentation of point clouds, which is modified and adapted in this study to detect concrete surface defects. This algorithm is originally designed to detect indoor building elements. The semantic segmentation of DGCNN is adapted to detect surface defects using point cloud datasets from scanning concrete bridge surfaces, as will be explained in Section 4. DGCNN, rather than operating on individual points, iteratively performs convolution on edges associating the neighborhood point pairs. As the edge is an important feature of the surface defects (e.g. cracks), using a deep learning method that can consider the edge feature and the relationship between neighboring points can improve the learning model's accuracy and efficiency.

The section starts with an adapted DGCNN, and then the main network of this research, which is the NVE-DGCNN is investigated in the next step. Furthermore, this section also proposes a method that includes post-processing of semantic segmentation results for the semi-automated as-inspected modeling purpose. The results of the defect semantic segmentation will be used to locate the defects in the BIM Model. This section focuses on the conceptual parts of the proposed methodology. More details about data collection and pre-processing are explained in Section 4. Section 5 gives the details of the implementation of the modified DGCNN models, while Section 6 shows the case study of as-inspected modeling.

3.1. Modified DGCNN models

(a) Adapted DGCNN: The following modifications have been done:

- **Network input parameters:** The network input parameters such as the class number, number of points per block, block size, and stride are modified based on annotated segments, sizes of the structural defects, and density of segmented parts. Wang et al. [19] used the block size of 1 $m \times 1$ m on the XY surface for rooms with a height of 3 m to detect indoor building elements using DGCNN. The number of points of 4096 is used for their training process. This setting results in a very low density of points for detecting most types of defects in this study (e.g. medium-sized spalls). In the adapted DGCNN, the block size of 40 $cm \times 40$ cm is set based on the sizes of the structural defects in the dataset. Moreover, the density of points in each block is increased by raising the number of points as explained in Section 3.2.6.
- **KNN of the EdgeConv layer:** In this study, in contrast with the original DGCNN, using the normalized X, Y, and Z coordinates of points for the KNN of the EdgeConv layer was considered to be unsuitable because normalization can destroy the critical information about the depth in the Y direction, which is much smaller than X and Z coordinates. A test was performed based on the original DGCNN to examine the effect of using the normalized location values for the KNN, which will be shown in Section 5.1. The test result showed that the models' performance declined significantly by considering the normalized location values for KNN. Therefore, the KNN for the EdgeConv layer of the adapted DGCNN is modified to compute the KNN based on the XYZ coordinates.

Table 1
DGCNN, adapted DGCNN and NVE-DGCNN hyperparameters.

Parameter	DGCNN	Adapted DGCNN	NVE-DGCNN
Classes	Building indoor objects (11 classes)	Cracks, spalls, no defect	Cracks, spalls, no-defect
Input Variables	*X, Y, Z, R, G, B, X', Y', Z'*	*X, Y, Z, R, G, B, Y'*	*X, Y, Z, R, G, B, Nx, Ny, Nz, Y'*
Number of points in each block	4096 *pts*	8192 *pts*	8192 *pts*
Size of blocks (*m*)	1 *m* × 1 *m* × Z_{max}	0.4 *m* × Y_{max} × 0.4 *m*	0.4 *m* × Y_{max} × 0.4 *m*
Stride	N.A.	0%	25%
Convolving direction	*XY* surface	*XZ* surface	*XZ* surface
Number of nearest neighbors (*k*)	20	20	20
Number of epochs	100	50	50
Optimizer	Adam	Adam	Adam
Weight vector for loss function	Softmax cross entropy	Weighted Softmax cross entropy	Weighted Softmax cross entropy
Learning rate	1e-3 (decays exponentially to a minimum of 1e-5)		

- **Convolving direction:** The *X*-axis is set along the concrete surface, the *Z*-axis is set in the vertical direction of the canonical coordinate system, and the *Y*-axis is set perpendicular to the surface and in the direction of the depth of the defects. The depth of defects is set to have positive *Y* values.
- **Modification related to the normalized coordinates:** The input point variables are changed from a 9-dimensional vector (*XYZ*, *RGB*, and *X'Y'Z'*) to a 7-dimensional vector (*XYZ*, *RGB*, and *Y'*) by removing the normalized values of the *X* and Z coordinates, and this vector is fed to the network. *X'*, *Y'*, and *Z'* are the normalized coordinates of *X*, *Y*, and *Z*, which have values between 0 and 1. As the depth of defects are in the *Y* direction, therefore this value can help the model to consider another feature related to the depth of defects.
- **Loss function:** As the defects' number of points in this research is less than the no defect number of points (defect points are almost 14% of the whole point cloud), which is known as the issue of *imbalanced datasets*, a weighted softmax cross entropy loss function is utilized to adapt the DGCNN models to the prepared dataset, and the corresponding weight vector is set based on the distribution of points label among the three classes. The method of distribution-based loss function is selected as a label weight score of classes is not dependent on how the instances are sampled, which can be more practical in this case. The label weight score of class *i* is calculated statically and added to the model using Eq. 1 and Eq. 2 [66,67].

$$\text{Distribution of points with class}_i \text{ label} = \frac{\sum_{i=1}^{N} C_i}{C_i} \tag{1}$$

$$\text{Label weight score of class}_i = \frac{\text{Distribution of points with class}_i \text{ label}}{\sum_{i=1}^{N} \text{Distribution of points with class}_i \text{ label}} \tag{2}$$

Where *N* is the number of classes and C_i is the total number of points for each class.

(b) NVE-DGCNN: The only modification in addition to those in adapted DGCNN is the consideration of the normal vector in NVE-

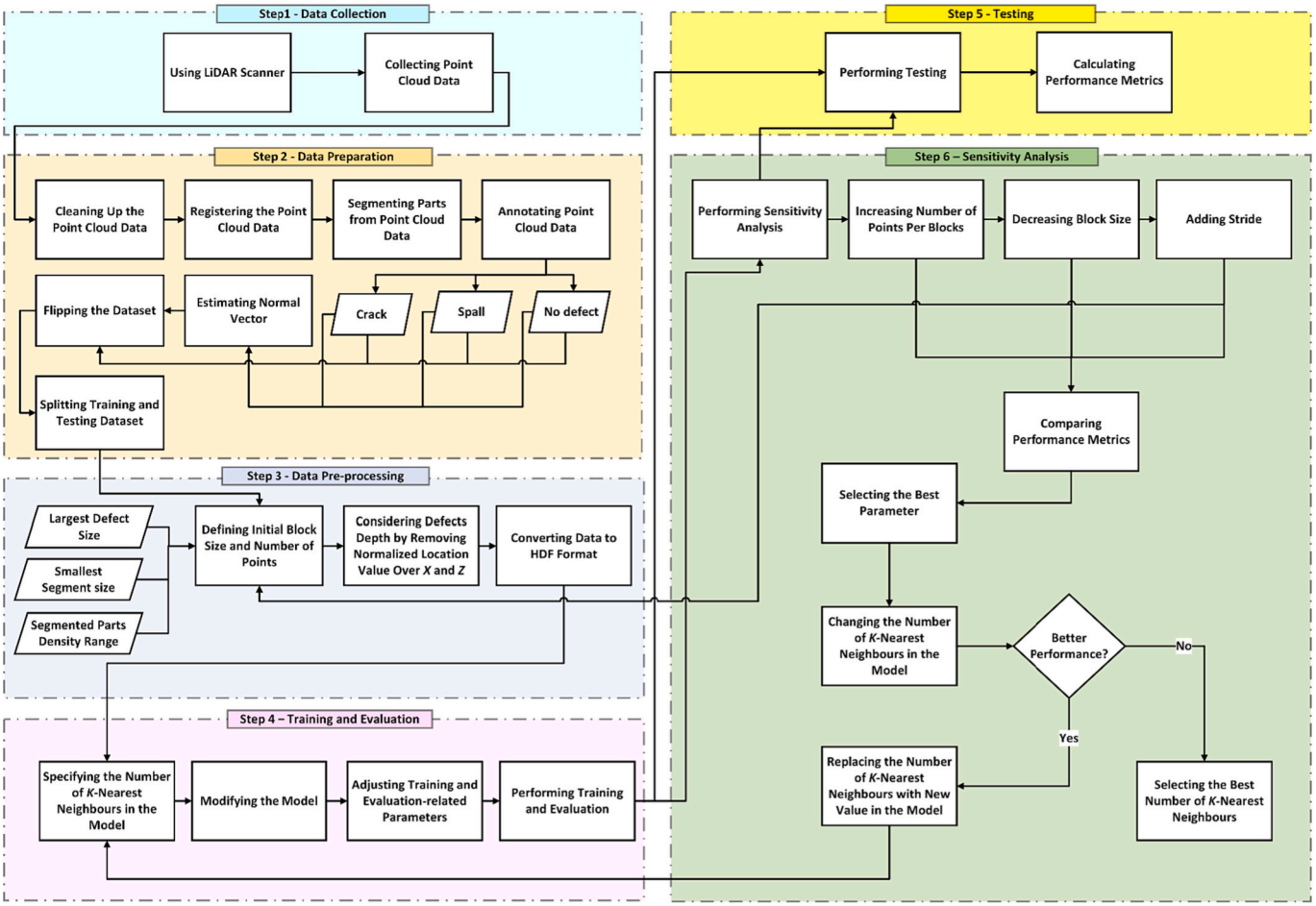


Fig. 2. Proposed method for concrete surface defect semantic segmentation using adapted DGCNN and NVE-DGCNN.

DGCNN. The adapted DGCNN method was enhanced by considering the normal vector feature in the NVE-DGCNN. NVE-DGCNN is modified to consider a 10-dimensional vector by adding the components of the normal vector (*Nx*, *Ny*, and *Nz*) as additional point features to the 7-dimensional vector in adapted DGCNN. The input variables are *XYZ*, *RGB*, *NxNyNz,* and *Y′*. Fig. 1 shows the architecture of the NVE-DGCNN. In the NVE-DGCNN segmentation model, the parameter *K* represents the number of edge features computed for every point within each EdgeConv layer, taking into account an input of *n* points. The model architecture encompasses a sequence of three EdgeConv layers, designed to capture and extract local edge features. Following these layers, a max-pooling layer is employed to aggregate global features from the block. Subsequently, three fully connected (FC) layers are utilized, incorporating a dropout layer in the middle, to classify the n points into three distinct classes. A weighted softmax cross entropy loss function is employed to optimize the classification performance, allowing for the incorporation of class-specific weights.

The adapted DGCNN and NVE-DGCNN hyperparameters are shown in Table 1.

3.2. Steps of applying the modified DGCNN models

There are six main steps in applying the modified DGCNN models: (1) data collection, (2) manual annotation, (3) data pre-processing, (4) training and evaluation, (5) testing, (6) sensitivity analysis. Fig. 2 shows the proposed method for concrete surface defect semantic segmentation using adapted DGCNN and NVE-DGCNN in detail. However, the normal vector estimation and sensitivity analysis steps are only implemented for NVE-DGCNN.

3.2.1. Data collection

The geometric features of defects, particularly the depth, play a significant role in extracting important features and having accurate results. Currently, most of available online datasets for concrete surface defects are image-based. Therefore, data collection is an important step and has to be done accurately. The scanner position and the scanning parameters, such as resolution, quality, Field of View (FOV), and the number of scanned points, are the factors that can affect the visibility of defects in the collected point cloud data.

3.2.2. Data preparation

After data collection, irrelevant points of the point cloud data in each scan need to be eliminated, and all the scans will be prepared for registration. The irrelevant points elimination process involves the removal of those data points that are considered unwanted or extraneous resulting from the scanning process. Then, different areas are cut from the registered point cloud data, and different parts are segmented in each area. The selected parts need to be manually annotated based on the types of targeted surface defects. As shown in Fig. 2, in this research, two main types of surface defects, which are cracks and spalls, are considered. Each part of the dataset is annotated into three categories of cracks, spalls, and no defect. Furthermore, in this research, to enlarge the size of the dataset, the augmentation method of flipping the point cloud is used, where the annotated parts are flipped with respect to the *YZ* plane. The authors developed a Python code to facilitate the flipping of annotated parts. This process involved reorienting the parts to achieve a flipped orientation.

3.2.3. Data pre-processing

This research considers two approaches for preparing the dataset and feeding the MLP classifier of the DGCNN network. In the first approach, the original dataset files are converted into data label files, which are *2D* matrices with *XYZRGBL* in each line. Then, each part is split into blocks, and for each block, normalized location values on the *Y* surface are added [19]. Each point is represented as a 7-dimensional vector of *XYZ*, *RGB*, and *Y′*. *XYZ* refers to points coordinate values in point cloud data.

Table 2
Model performance metrics.

Performance metrics	Equation
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1 score	$2\times\frac{Precision\times Recall}{Precision+Recall}$
Intersection over Union (IoU)	$\frac{TP}{FP+TP+FN}$
Overall accuracy	$\frac{TP_{Crack}+TP_{Spall}+TP_{No\ defect}}{All\ perdiction\ of\ the\ points}$

Note: *TP* refers to true positives, *FP* refers to false positives, and *FN* refers to false negatives

RGB (Red, Green, and Blue) refers to the color of each point in the point cloud. *L* refers to the annotated label for each point (cracks, spalls, and no defect). These features were used for the training process of the adapted DGCNN. In the second approach, an additional hand-crafted point feature, which is the normal vector (*Nx, Ny,* and *Nz*) is added to feed a MLP classifier of NVE-DGCNN. Previous works, such as Hyeon et al. [68] investigated the effect of considering normal vectors as an additional feature for semantic segmentation of building elements and stated that considering the normal vectors in CNN networks can improve the model's performance.

Then, the sizes of blocks are defined based on the sizes of the structural defects (smallest segment or the largest defect) in the dataset. In this research, the smallest dimension of segments is 46 *cm*, and the largest defect size is 60 *cm*. Hence, the selected block size in the data pre-processing step is assumed to be at least 40 *cm* × 40 *cm* on the *XZ* surface, with the depth of the defects as the third dimension, which is equal to the depth of the deepest defect in each segment. As shown in Fig. 2, in this step, the wrapped and normalized points inside the blocks are converted to Hierarchical Data Format (HDF) [69], and HDF5 files are used for the training process in the next step.

3.2.4. Training and evaluation

After the data pre-processing step, the dataset will be fed as input for the modified model. Each CNN layer receives inputs through the previous layer's local receptive fields. Neurons extract basic geometric features such as edges, boundaries, and corners using local receptive fields, which are also known as filters or kernels. While the model is being trained, about 20% of the data is used to assess initial accuracy, observe how the model learns, and adjust hyperparameters. As discussed in Section 3.1, a series of three EdgeConv layers followed by three fully-connected layers are included in the segmentation model of DGCNN, and the number of the *K*-nearest neighbors of a point for EdgeConv layers is specified for the input of *n* points in the model. The number of the *K*-nearest neighbors of a point for EdgeConv layers is set equal to 20 following the suggested value by Wang et al. [19]. The details of the training and evaluation are given in Section 5.

3.2.5. Testing

To validate the model accuracy, the unseen parts of the dataset, which are not used in the training and evaluation steps, are used for the testing step. The confusion matrix is used to describe the model's performance using the equations presented in Table 2. In this research, the term *overall accuracy* refers to the percentage of correct predictions for the test data. Furthermore, the *recall* is assumed to be more relevant than precision as the process of concrete surface inspection aims to minimize the chance of missing actual defect points, which can be achieved by minimizing the *False Negative* prediction of the model.

3.2.6. Sensitivity analysis

As shown in Fig. 2, sensitivity analysis was done in this research to investigate the effect of different input variables on the network's

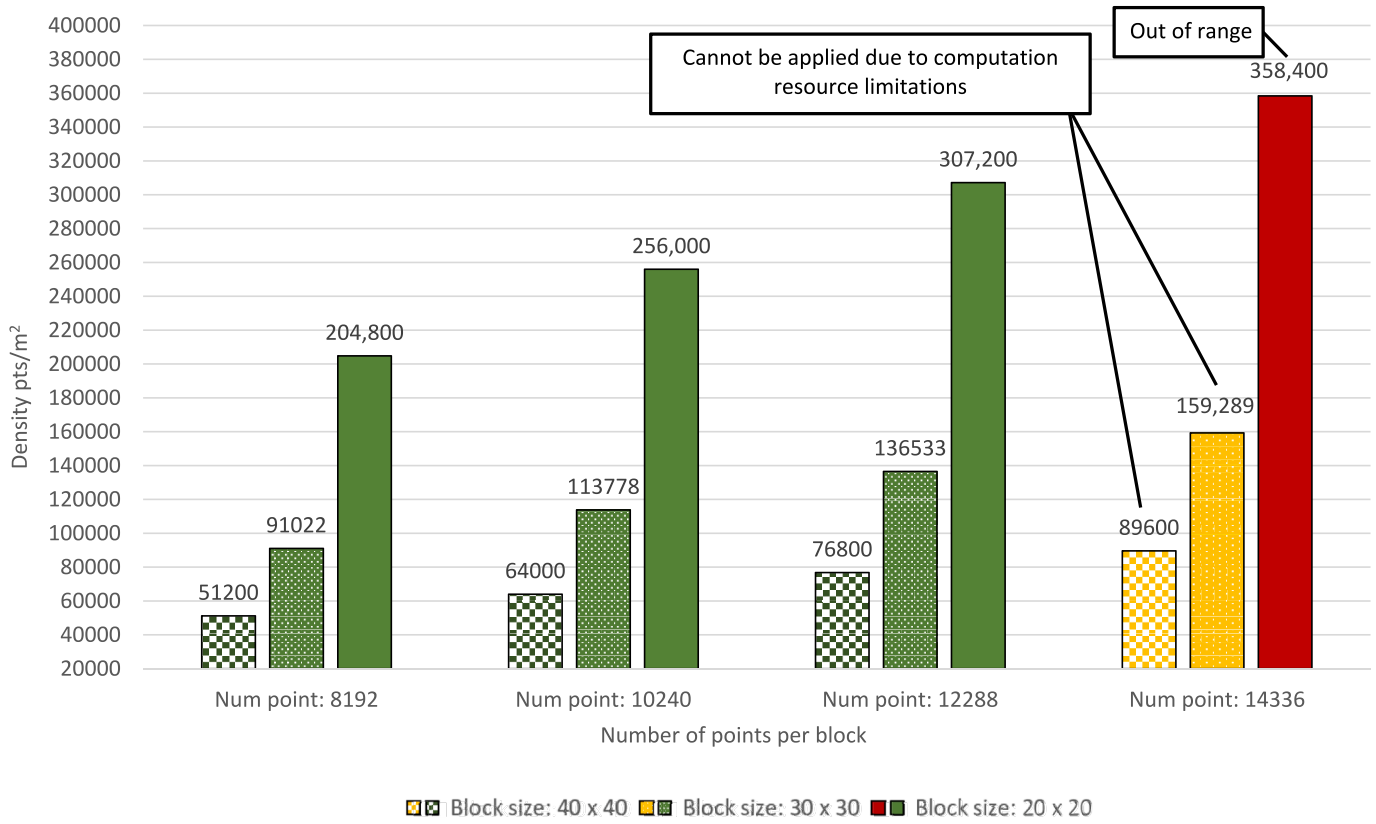


Fig. 3. List of number of points per block, block sizes (*cm*), and their densities.

performance. The dataset's density depends on the block size and the number of points per block. Blocks with densities more than the pre-set value result in up-sampling. On the other hand, densities less than the pre-set value for each block result in down-sampling. Therefore, the distribution of segments based on their densities should be considered. The hyperparameters related to the dataset, which are considered in this research, are: (1) number of points, (2) size of the block, and (3) size of stride. The number of points per block is selected based on the density range and the pre-set default block size (*40 cm* × *40 cm*). The number of points per block of 8192 is selected as the first acceptable value. Then the density of points in each block is increased by raising the number of points. The density of most segments of the prepared dataset is between 9049 and 329,369 pts/m^2. Fig. 3 shows the list of number of points per block, block sizes, and their densities. Increasing the number of points to more than 12,288 could not be applied due to computation resource limitations (i.e., RAM). Moreover, the density of the block with the size of *20 cm* × *20 cm* and number of points of 14,336 is more than 329,369 pts/m^2, which is out of the range for this research. Then the block size is decreased for the same pre-set number of points to increase the density of points in each block. Stride can be added to shift the number of points per block over the input matrix. Considering the overlapped number of points per block may improve the results. Therefore, the effect of stride on the results is considered in the sensitivity analysis.

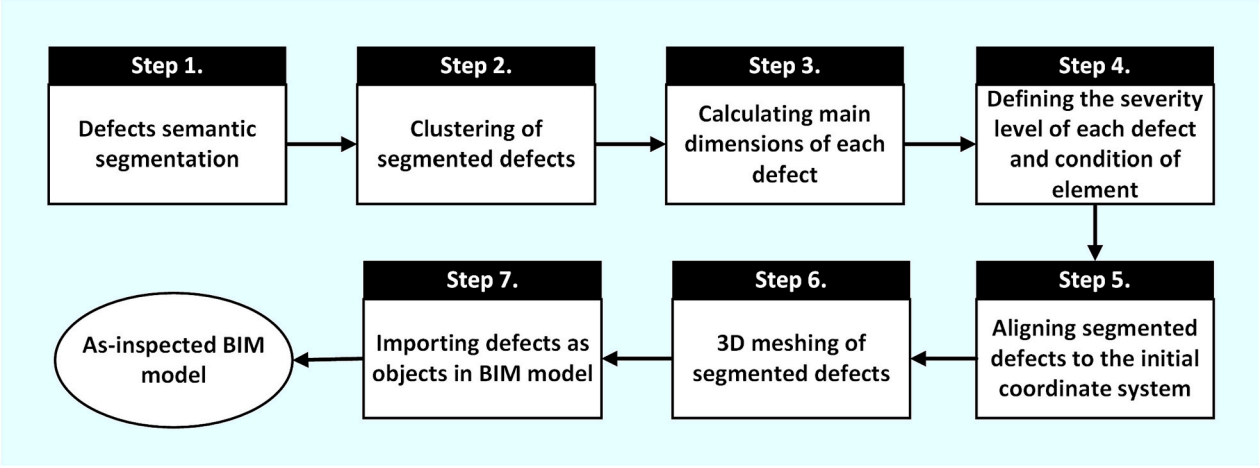


Fig. 4. Workflow of as-inspected modeling.

Table 3
Severity of crack and spall defects based on OSIM [74].

Surface defect types	Severity (all dimensions in *mm*)			
Crack	Hairline (width < 0.1)	Narrow (0.1 ≤ width ≤ 0.3)	Medium (0.3 < width ≤ 1.0)	Wide (1.0 < width)
Spall	Light (Any direction <150 or depth < 25)	Medium (150 ≤ Any direction ≤300 or 25 ≤ depth ≤ 50)	Severe (300 < Any direction ≤600 or 50 < depth ≤ 100)	Very severe (600 < Any direction or 100 < depth)

3.3. As-inspected modeling

The current BIM information does not support the representation and integration of defect information. Integrating detected defect information with BIM will facilitate accessing and updating the inspected defect information at different phases of the lifecycle resulting in improved efficiency and reduced rate of data input errors. As-inspected modeling will help to store the inspection in an efficient and precise way. It can also enable the tracking and analysis of the changes throughout the lifecycle. The as-inspected BIM model not only contains the basic geometry of defects but also semantic information about their type, severity, etc. Fig. 4 shows the workflow of the as-inspected modeling process. The following main steps are used to semi-automate the process of as-inspected modeling:

(1) Defects semantic segmentation: The as-built bridge model is assumed to be available. The semantic segmentation results of NVE-DGCNN for crack and spall defects are used for the as-inspected modeling purpose.

(2) Clustering of segmented defects: Density-based spatial clustering, which is proposed by Liu et al. [70], is used in this step to create spatial proximity relationships to cluster the defects [71]. The density-based spatial clustering algorithm uses the concept of reachability, which refers to how many neighbors a point has within a given radius. This algorithm does not need a specific number of clusters and works well for noisy datasets.

(3) Calculating main dimensions of each defect: An algorithm is used to find the Minimum Bounding Box (MBB) for each cluster [72,73], and geometrical information of cracks and spalls including the defects' length, width, and depth are calculated based on the Euclidean distance between the corners of the associated bounding box.

(4) Defining the severity level of each defect and condition of element: The severity level of each defect is defined based on Table 3. This table shows the severity levels of crack and spall defects based on the Ontario Structure Inspection Manual (OSIM) [74]. The element condition is also defined based on the severity level, as discussed in OSIM.

(5) Aligning segmented defects to the initial coordinate system: The segmented defects are aligned to the initial coordinate system by considering each point's normal vector using the Normal Iterative Closest Point (NICP) algorithm [75], which is proposed by Serafin and Grisetti [76]. NICP is an iterative technique that leverages the normal distribution transform to generate an initial estimation of the relative transformation between two point clouds. This algorithm iteratively refines the transformation by aligning the corresponding points in the source and target point clouds, with the aim of minimizing the overall registration error. By utilizing the normal distribution transform, which characterizes the local surface geometry, NICP enables the algorithm to effectively handle complex and non-rigid deformations in the point cloud data. Although some tools such as CloudCompare software can be used to align the point clouds manually by picking at least four pairs of reference points in the source and target point clouds, this process is time-consuming.

(6) 3D meshing of segmented defects: In this step, the clusters of detected defects are converted into a 3D mesh product using 3DReshaper software [77] to have an accurate model of the defect objects.

(7) Importing defects as objects in BIM model: In this step, a Dynamo script using Mesh Toolkit [78] is utilized to import the 3D mesh defects into the BIM model. Although the case study that will be explained in Section 6 is about bridge inspection, where the structure model is referred to as Bridge Information Model (BrIM), the term BIM will be used in the rest of this paper.

4. Data collection and dataset pre-processing

This study used point cloud datasets from four reinforced concrete bridges in Montreal, scanned using a FARO Focus3D scanner [79]. The specifications of this scanner are presented in Table 4. The images of the scanned bridges are shown in Fig. 5. The position of the scanner was at a distance of 5 to 10 m from the scanned surfaces. Table 5 shows the scanning parameters. CloudCompare software [80] is used to register and eliminate the irrelevant points of the point cloud data. The scanned data's quality depends on the two main parameters of density and accuracy [81]. The number of points in a specific area represents the point cloud density [82]. The resolution parameter represents the number of points that the scanner uses to measure the environment during the scanning process (between *1* (710.7 million points) - *1/32* (11.1 million points)), and the quality represents the number of times the scanner hits the same point during the scanning (between *1*× - *8*×) [83]. Therefore, the distance between two points next to each other depends on the resolution parameter.

The scanning process in this step is affected by several factors, such as the battery capacity and performance limitations, especially in severe weather conditions, scanning time, and traffic constraints. For this reason, different settings, including different numbers of stations, were used to scan each of the bridges. In some scans, the FOV was reduced to avoid scanning irrelevant objects (e.g. moving vehicles).

A careful selection process was conducted within each scanned area to identify several parts that exhibited defects. Subsequently, within each selected part, a comprehensive annotation procedure was undertaken to categorize the specific instances of cracks, spalls, and no defect regions. Careful consideration was given to selecting the most optimal scans for this analysis, ensuring that the quality of the available scans from the scanned surfaces was of the highest standard. Ultimately, a total of 102 selected segmented parts were chosen based on the aforementioned criteria, facilitating a comprehensive examination and analysis of the identified defects in the subsequent stages of the study. The number of annotated cracks in the selected parts is 595, and the number of annotated spalls is 773. The annotation process is done manually in CloudCompare software using the following rules based on experience: (1) a specific range of 150,000 pts. to 400,000 pts. is considered for the number of points of each selected part; (2) the scanned surfaces are classified into rectangular parts because of the box

Table 4
FARO Focus3D LiDAR scanner specifications [83].

LiDAR	Points per Second	Field of View		Angular Resolution	Accuracy	Measurement Range
		Vertical	Horizontal			
FARO Focus 3D	976,000	305°	360°	0.009°	±2 *mm*	1.5 *m* – 120 *m*

(a) Bridge 1: Guy Street

(b) Bridge 2: Lucian L'Allier Street

(c) Bridge 3: Avenue Atwater

(d) Bridge 4: Chemin Macdonald

Fig. 5. Scanned bridges.

Table 5
Scanning parameters of four scanned bridges in Montréal.

Scans		Number of Stations	Resolution	Quality	Horizontal FoV	Vertical FoV	Number of Points (Mpts)
Bridge 1	Scan 1	8	1/4	6×	23° to 259°	−42.5° to 71°	25.5
	Scan 2	4	1/4	6×	23° to 259 °	−42.5° to 71°	25.5
Bridge 2	Scan 3	6	1/1	2×	0° to 360°	−60° to 90°	710.7
Bridge 3	Scan 5	4	1/2	4×	0° to 360°	−45° to 71°	134.5
Bridge 4	Scan 6	2	1/2	4×	0° to 360°	−60° to 90°	177.7

Table 6
The statistics of the prepared dataset.

Dataset		Number of segmented parts	Number of points	Defects				No defect
				Cracks		Spalls		Number of points
				Number of cracks	Number of points	Number of spalls	Number of points	
Training (59.5%)	Area 1	32	10,418,902	264	104,256	226	715,768	9,598,878
	Area 2	44	11,003,768	334	112,436	266	282,822	10,608,510
	Area 3	42	10,651,316	160	67,714	356	744,356	9,839,246
Evaluation (19.6%)	Area 4	44	10,552,584	192	80,454	328	762,156	9,709,974
Testing (20.9%)	Area 5	42	11,257,240	240	128,538	370	1,365,228	9,763,474
Total		204	53,883,810	1190	493,398	1546	3,870,330	49,520,082

shape of the blocks in the model; and (3) the part size should consider the higher density of points in some parts and it should not contain more than the maximum defined number of points, which is 400,000 pts. To achieve this, all areas, excluding the surface edges, were carefully delineated. Each area was then further segmented into individual parts. The specific focus of the segmentation was on identifying and distinguishing two types of defects, which are cracks and spalls. Furthermore, the handcrafted normal vector feature (*Nx, Ny,* and *Nz*) is computed in CloudCompare software to prepare the NVE-DGCNN dataset. The annotated datasets are split into five areas. Areas 1 to 3 are used for training, Area 4 is used for evaluation, and Area 5 is dedicated to testing. The total number of segmented parts after adding the flipped data is 204 parts. The statistical information of the dataset, including the flipped data, is given in Table 6. Fig. 6 shows the structure of preparing the dataset including a sample of an annotated segment, where the light-gray background represents no defects, black areas represent spalls, and dark-gray thin areas represent cracks.

5. Implementation of the modified DGCNN models

5.1. Implementation of adapted DGCNN

A Compute Canada cluster is used to implement this case study using 4 NVIDIA V100 Volta GPUs with 32 GB RAM per GPU, 24 CPUs, and 123 GB of memory. The number of epochs is set to 50. The initial learning rate is 0.001, and it decays exponentially to a minimum of 1e-5. The percentage of the defect points is almost 14% of the whole point cloud and is much less than the no-defect points (86%). Therefore, as discussed

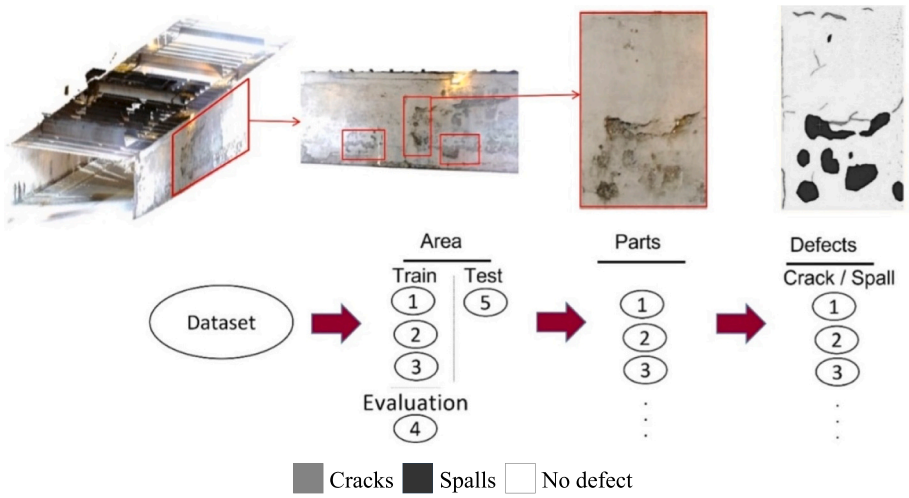


Fig. 6. The structure of the dataset

Table 7
Training and evaluation results for adapted DGCNN.

Case	Number of sampled points for each block	Block size (*cm*)	Training		Evaluation		Training time
			Mean loss	Overall accuracy (%)	Mean loss	Overall accuracy (%)	
A1	8192	40 × 40	0.0022	97.54	0.0081	97.50	13 h 44 m
B1	10,240	40 × 40	0.0024	97.39	0.0090	97.65	16 h 35 m
C1	12,288	40 × 40	0.0030	97.04	0.0082	96.88	20 h 18 m

Table 8
Testing results for adapted DGCNN (%).

Case	Overall accuracy	Cracks				Spalls				No defect			
		Precision	Recall	F1 score	IOU	Precision	Recall	F1 score	IOU	Precision	Recall	F1	IOU
A1	95.94	69.98	55.20	61.76	44.68	79.30	89.77	84.2	72.72	98.54	97.17	97.85	95.79
B1	95.59	68.95	55.31	61.38	44.28	77.47	89.41	83.0	71.0	98.48	96.82	97.64	95.39
C1	95.24	49.73	58.67	53.83	36.83	77.00	87.40	81.9	69.3	98.48	96.64	97.55	95.22

Table 9
Defect semantic segmentation recall based on the depth of defects for adapted DGCNN (%).

Case	Number of sampled points for each block	Depth (*cm*)					
		$D \leq 3$		$3 < D < 7$		$7 \leq D$	
		Cracks	Spalls	Cracks	Spalls	Cracks	Spalls
A1	8192	35.22	90.78	44.87	87.59	76.91	92.68
B1	10,240	36.65	88.48	42.52	86.99	79.00	93.22
C1	12,288	39.22	81.91	48.07	84.39	80.04	93.33

in Section 3.1, a weighted softmax cross entropy loss function is defined in the model based on the points distribution of the classes (cracks, spalls, and no defect), which is [0.714, 0.271, 0.016]. By using a weighted loss function, the effective weight of points of each class in the correcting process of backpropagation can be adjusted.

In the first step, to implement the adapted DGCNN, three cases are defined with different numbers of input points of 8192, 10,240, and 12,288 (Case A1 to C1), which are sampled for each block during the training process. The training and evaluation results, including the overall accuracy and mean loss of Cases A1 to C1, are presented in Table 7. Precision, recall, F1 score, IoU, and overall accuracy are calculated to evaluate the semantic segmentation results for Cases A1 to C1. The test results of the adapted DGCNN (Table 8) show that the detecting recall for cracks and spalls for Case C1 (12,288 points) are 58.67% and 87.40%, respectively. Increasing the number of points from 8192 to 12,288 improved the cracks semantic segmentation recall from 55.20% to 58.67%. However, this increase resulted in decreasing the spall recall from 89.77% to 87.40%, and no defect recall from 97.17% to 96.64%. This is because increasing the number of points sometimes can cause overfitting [84].

As the depths of segmented parts are different, and the learning process depends on the maximum depth of the part's defects, the recall result of the tests is categorized based on the depth of segmented parts used in the test as shown in Table 9. As shown in this table, deeper parts can increase recall up to 80.04% for cracks and 93.33% for spalls.

Furthermore, as explained in Section 3.1, a test was performed based on the original DGCNN to examine the effect of using the normalized *X, Y, and Z* values for the KNN. The test was performed for the number of points of 8192, and block size of 40 × 40 *cm* with no stride. The comparison of the DGCNN with original KNN and adapted DGCNN with modified KNN shows that the models' performance declined significantly by considering the normalized location values for KNN. The recall results of DGCNN with original KNN were 41.17% for cracks and 41.27% for spalls. The recall results of adapted DGCNN with modified KNN were 55.20% for cracks and 89.77% for spalls.

Three samples of the test results for Case C1 from the adapted DGCNN are shown in Fig. 7. The light-gray background represents no defect, black areas represent spalls, and dark-gray thin areas represent cracks.

This part of the implementation defines fifteen cases based on the

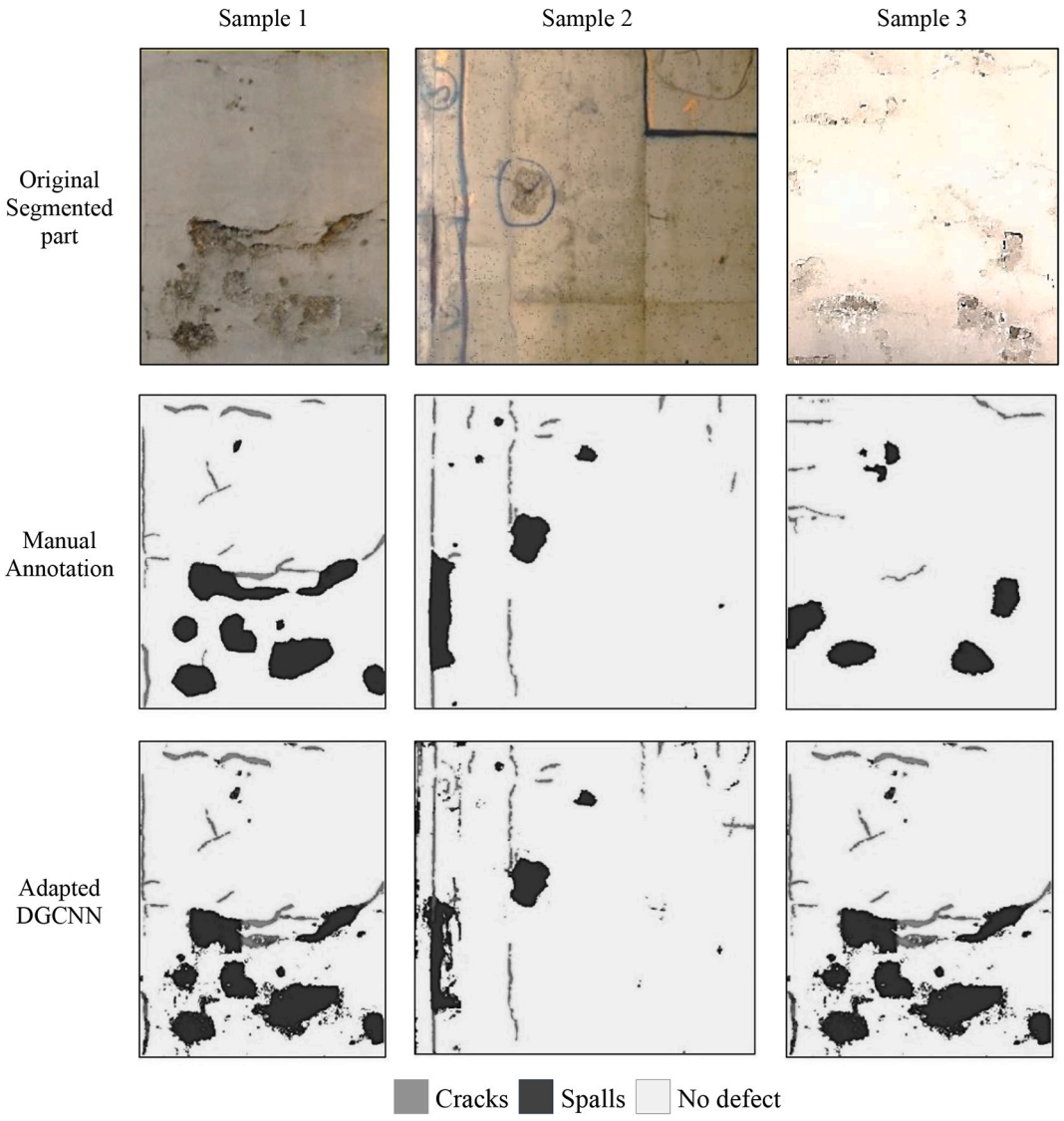


Fig. 7. Test results from three samples of adapted DGCNN (Case C1).

Table 10
Training and evaluation results for NVE-DGCNN.

Case	Number of points per block	Block size (*cm*)	Stride (*cm*)	Training		Evaluation		Training time
				Mean loss	Overall accuracy (%)	Mean loss	Overall accuracy (%)	
A2	8192			0.0012	98.60	0.0097	98.58	14 h 46 m
B2	10,240	40 × 40	40 (0%)	0.0008	99.12	0.0077	98.36	18 h 35 m
C2	12,288			0.0009	98.76	0.0070	98.62	20 h 26 m
D2	8192			0.0013	98.74	0.0092	95.16	21 h 01 m
E2	10,240	30 × 30	30 (0%)	0.0007	99.13	0.0104	95.50	27 h 05 m
F2	12,288			0.0013	98.67	0.0069	95.26	31 h 46 m
G2	8192			0.0011	98.97	0.0090	94.84	41 h 44 m
H2	10,240	20 × 20	20 (0%)	0.0008	99.14	0.0117	98.14	52 h 50 m
I2	12,288			0.0016	98.46	0.0148	97.58	65 h 14 m
J2	8192			0.0004	99.53	0.0064	97.47	19 h 50 m
K2	10,240	40 × 40	30 (25%)	0.0006	99.34	0.0077	98.05	24 h 35 m
L2	12,288			0.0006	99.29	0.0087	96.63	29 h 4 m
M2	8192			0.0003	99.62	0.0084	97.88	36 h 41 m
N2	10,240	40 × 40	20 (50%)	0.0003	99.62	0.0091	97.56	44 h 55 m
O2	12,288			0.0004	99.59	0.0080	97.28	55 h 31 m

NVE-DGCNN to identify the effect of the hyperparameters on the performance.

The training and evaluation results, including the overall accuracy and mean loss of Cases A2 to O2, are presented in Table 10. Precision, recall, F1 score, IoU, and overall accuracy are calculated to evaluate the testing of the cases as shown in Table 11. The processing time for testing

Table 11
Testing results for NVE-DGCNN (%).

Case	Number of points per block	Block Size (*cm*)	Stride (*cm*)	Overall accuracy	Cracks				Spalls				No defect			
					Precision	Recall	F1 score	IOU	Precision	Recall	F1 score	IOU	Precision	Recall	F1 score	IOU
A2	8192	40 × 40	40 (0%)	98.28	92.71	96.20	94.42	89.43	89.97	95.30	92.56	86.15	99.44	98.67	99.06	98.13
B2	10,240			98.34	92.97	95.49	94.22	89.06	90.95	94.60	92.74	86.46	99.36	98.83	99.09	98.20
C2	12,288			98.13	93.75	95.32	94.53	89.63	88.57	95.53	91.92	85.04	99.47	98.48	98.97	97.97
D2	8192	30 × 30	30 (0%)	98.12	94.16	97.57	95.84	92.01	88.95	94.91	91.83	84.90	99.39	98.52	98.95	97.93
E2	10,240			97.96	96.49	94.94	95.71	91.76	90.11	91.53	90.81	83.17	98.96	98.79	98.87	97.77
F2	12,288			98.29	95.43	97.78	96.59	93.40	83.76	95.31	89.16	80.44	99.64	98.53	99.08	98.18
G2	8192	20 × 20	20 (0%)	93.14	96.26	95.47	95.86	92.05	62.69	92.99	74.89	59.86	99.08	93.13	96.01	92.33
H2	10,240			97.54	93.31	97.52	95.37	91.15	88.52	89.53	89.02	80.22	98.72	98.53	98.63	97.30
I2	12,288			97.07	97.38	90.36	93.74	88.21	87.81	85.32	86.55	76.29	98.18	98.61	98.39	96.84
J2	8192	40 × 40	30 (25%)	98.88	95.98	98.56	97.25	94.65	95.55	96.50	96.02	92.35	99.46	99.27	99.36	98.73
K2	10,240			98.17	93.67	97.64	95.61	91.60	92.83	94.18	93.50	87.79	99.11	98.83	98.97	97.95
L2	12,288			98.34	96.72	95.87	96.29	92.85	91.35	94.52	92.91	86.76	99.30	98.86	99.08	98.18
M2	8192	40 × 40	20 (50%)	98.42	97.32	97.68	97.50	95.13	93.15	93.78	93.47	87.74	99.16	99.06	99.11	98.24
N2	10,240			98.44	96.41	97.23	96.82	93.84	94.10	92.92	93.51	87.80	99.06	99.22	99.14	98.29
O2	12,288			98.47	96.93	94.02	95.45	91.30	93.92	93.22	93.57	87.91	99.10	99.24	99.17	98.36

the dataset in the inspection of one defect is less than one minute. The results show that decreasing the block size to 20 × 20 *cm* or adding the 50% stride for the block with the size of 40 × 40 *cm* will decrease the cracks and spalls recall. The best test results of the NVE-DGCNN show that the recall for cracks and spalls for Case J2 (8192 points and 25% stride) are 98.56% and 96.50%, respectively.

For sensitivity analysis of NVE-DGCNN, fifteen cases (Case A2 to O2) are defined and validated using three numbers of points of 8192, 10,240, and 12,288, and three block sizes of 40 × 40 *cm*, 30 × 30 *cm*, and 20 × 20 *cm*. Moreover, the stride could not be applied to 20 × 20 *cm* due to the computation resource limitations (i.e., RAM size). Therefore, the effect of stride size is investigated for the block size of 40 × 40 *cm* by applying 0%, 25%, and 50% strides (40, 30, and 20 *cm*). Nine cases were studied with 0% stride (A2 to I2). In all cases with the same block size and stride, the number of points increased from 8192 to 12,288 (Case A2 to O2). In three cases (J2 to L2), the stride was decreased by 25%, and in the remaining cases (M2 to O2), the stride considered was 50%.

Increasing the number of points will decrease the average recall of defects in the NVE-DGCNN network. Based on [19], this can be explained by the mismatch between the density and the value of the number of the *K*-nearest neighbors. Moreover, increasing the number of points may occasionally result in overfitting [84]. Decreasing the block size to 25% (30 *cm* × 30 *cm*) will increase the cracks average recall by 1.17%. However, the accuracy of spall decreased by 0.17%. The average recall of both cracks and spalls was decreased by decreasing the block size to 50% (20 *cm* × 20 *cm*). Decreasing the stride to 25% (30 *cm*) improved the cracks average recall by 1.63%. The average recall of both cracks and spalls was decreased by decreasing the stride to 50% (20 *cm)*.

NVE-DGCNN improves the semantic segmentation performance of cracks a little more than spalls (almost 2% in Case J2). Adding the normal vector feature to the points specifies additional geometric information. On the other hand, the range of the change of normal vector in cracks is less than in spalls. Therefore, considering the normal vector may increase the chance of detecting cracks more than spall. In the testing phase, most unforeseen cracks points were considered as no defect in adapted DGCNN, but where correctly classified as cracks in NVE-DGCNN.

Furthermore, the EdgeConv layer can detect the edges by applying an operation on edges to define the relationships between a point and its neighbors [19,85]. Three samples of the best results for Case J2 from the NVE-DGCNN are shown in Fig. 8. As shown in this figure, the NVE-DGCNN detected most of the cracks when the normal vector feature was added to the points. In addition, in Sample 2, the network efficiently detected a line that looks like a crack as a no defect. This example indicates the advantage of point cloud-based methods over image-based methods.

As shown in Table 12, deeper parts can increase recall up to 99.38% for cracks and 99.41% for spalls for Cases J2.

As shown in Table 13, the comparison between adapted DGCNN and NVE-DGCNN shows that using the normal vector as an additional point feature improved the model's accuracy for the same number of points of 8192, 10,240, and 12,288, and block size of 40 × 40 *cm* with 0% stride..

In addition, to determine the effect of *K*-nearest neighbors in the model, five cases with different numbers of *K* (5, 10, 15, 20, and 25) are defined for Case J2, which has the best performance. As Table 14 shows, the case of *K* equal to 20 (suggested value by Wang et al. [19]) still has the best performance, and increasing the number of *K* will decrease the network's performance. As explained earlier in this section, mismatch between the density and the value of the number of the *K*-nearest neighbors may decrease the performance [19].

6. Case study of as-inspected modeling

This section aims to automate the process of as-inspected modeling based on the results of the concrete surface defects semantic segmentation, including cracks and spalls. The goal of the case study is to

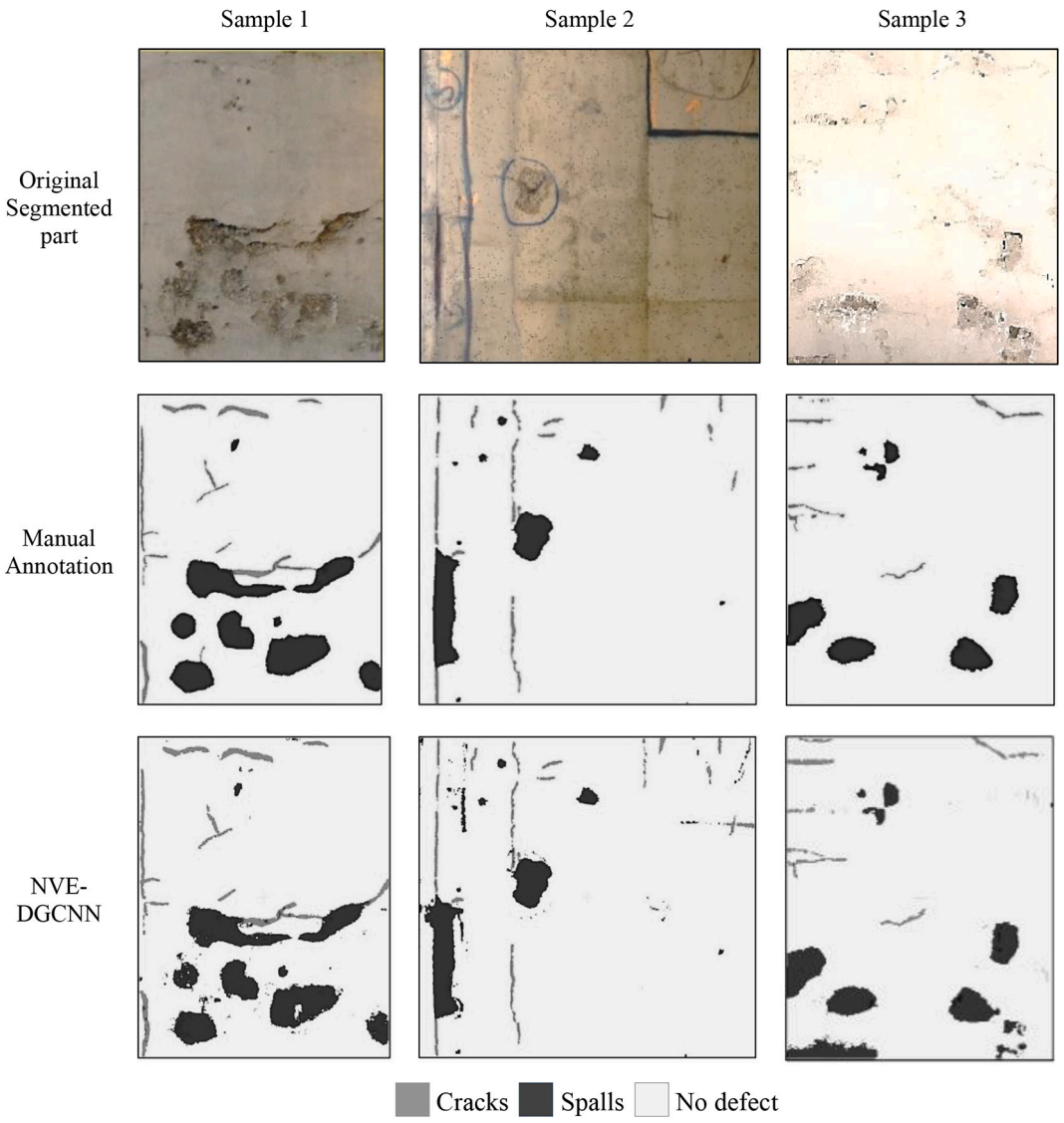


Fig. 8. Test results from three samples of NVE-DGCNN (Case J2).

Table 12
Semantic segmentation recall based on the depth of defects for NVE-DGCNN (%).

Case	Number of sampled points for each block	Depth (*cm*)					
		$D \leq 3$		$3 < D < 7$		$7 \leq D$	
		Cracks	Spalls	Cracks	Spalls	Cracks	Spalls
J2	8192	94.67	94.75	99.32	97.36	99.38	99.41

Table 13
Comparison of the results of adapted DGCNN and NVE-DGCNN (recall %).

Number of sampled points for each block	Block size (*cm*)	Adapted DGCNN		NVE-DGCNN	
		Cracks	Spalls	Cracks	Spalls
8192	40 × 40	55.20	89.77	96.20	95.30
10,240	40 × 40	55.31	89.41	95.49	94.60
12,288	40 × 40	58.67	87.40	95.32	95.53

Table 14
Results of best NVE-DGCNN case with different numbers of *K*-nearest neighbors.

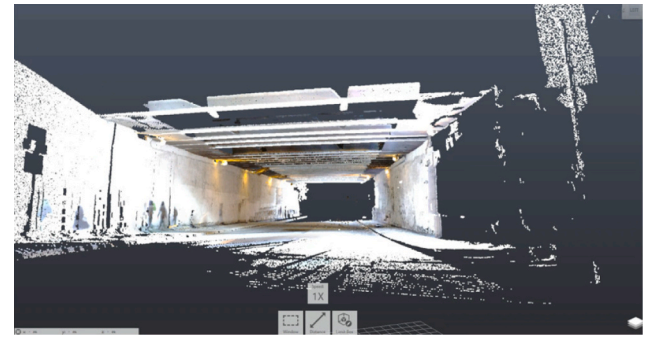
Case	Number of sampled points for each block	Stride (*cm*)	Number of nearest neighbors (*K*)	Cracks	Spalls
				Recall	Recall
J2–1	8192	30 (25%)	5	98.14	95.22
J2–2			10	97.67	94.23
J2–3			15	98.21	94.21
J2–4			20	98.56	96.50
J2–5			25	97.34	94.98

implement as-inspected modeling. The case study involves one of the scanned bridges in Section 4.4. The scan stations were located under the bridge on *Rue Lucien-L'Allier* between *Rue Saint-Antoin West* and *René-Lévesque Boulevard West, Montreal*. Fig. 9 shows the images of bridge.

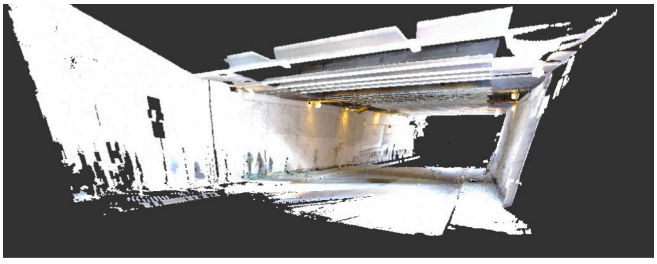
The point cloud registration process is done with Trimble RealWorks, and the point cloud of the bridge is cleaned up from most unrelated data using Recap Pro. Fig. 10(a) shows an instance of the raw point clouds in their initial state prior to undergoing the clean-up process. Fig. 10(b) shows the cleaned-up scan of the bridge.

The inspector usually focuses on scanning the areas that are expected to have defects. However, in this study, an additional step of scan-to-BIM

Fig. 9. Images of the bridge.



(a) Before cleaning



(b) After cleaning

Fig. 10. Cleaned up scan of the bridge.

was done, as the 3D model of the bridge was not available. In this step, the registered scan file was exported to ".rcs" format, and then imported into Autodesk Revit 2019 software. Fig. 11(a) shows the imported cloud data in Revit software. Fig. 11(b) shows the 3D model of the bridge based on point cloud data.

As Fig. 12 shows, a sample of the point cloud on the abutment surface of the bridge is selected for defect semantic segmentation.

Fig. 13(a) shows crack and spall defects on point cloud data. Fig. 13 (b) shows detected crack and spall defects using NVE-DGCNN. As discussed in Section 3.3, detected crack and spall defects' length, width, and depth were calculated using density-based spatial clustering and the MBB algorithm. Figs. 14(a) and (b) show the clusters of the cracks and spalls, respectively. Fig. 15(a) shows an example of the MBB of crack cluster Number 0 of in the sample. Fig. 15(b) shows an example of the MBB for spall cluster Number 2 in the sample. Tables 15 and 16 show the geometrical and semantic information of cracks and spalls of the sample, respectively.

The 3D model of the as-inspected bridge based on point cloud data is used for defect product modeling. After applying clustering and MBB algorithms, the segmented defects are aligned to the initial coordinate system using NICP algorithms, as explained in Section 3.3. Then, the segmented defects with an aligned coordinate system are saved as new files to be used in the 3D modeling step. The results of aligned segmented defects look similar to Fig. 12.

The clusters of detected defects are converted to 3D mesh products to have an accurate model of the defect objects. A Dynamo script using Mesh Toolkit [78] is utilized to import the 3D mesh defects into the BIM model. The average processing time for the 3D defects presentation in the BIM model is a few seconds. Fig. 16 shows the imported defect objects into the model of the bridge abutment in Revit. Finally, the semantic information of each defect, including the defect's type, severity levels, and the condition of the defected element, are added manually to the BIM model. Fig. 17 shows an example of semantic information for cluster Number 4 of spalls in the BIM model.

7. Discussion

In previous works, such as [63,64], it is evident that image-based processes have demonstrated the ability to provide visually appealing and informative visualizations of inspected defects. However, it is important to note that while these image-based processes excel in visualization, they often lack crucial depth information. The advantage of the post-processing stage lies in its ability to complement the visual representation of defects with additional depth information. By incorporating depth data, the post-processing stage enhances defect quantification, characterization, or spatial understanding, which is crucial for detailed inspection.

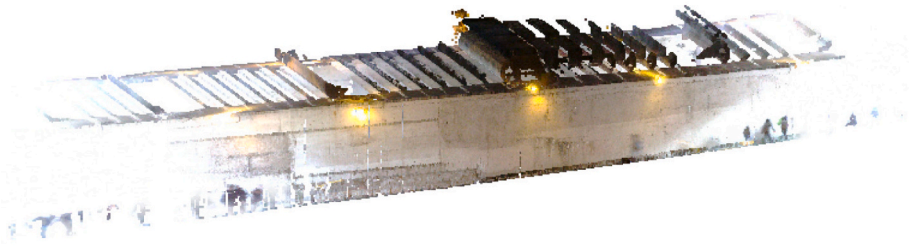
In another research, Bolourian et al. [86] investigated the effect of adding normal vectors to an adapted PointNet++ (SNEPointNet++). The best results of this network along with the best results of the NVE-DGVNN are shown in Table 17. Compared to SNEPointNet++, NVE-DGCNN recalls in detecting cracks and spalls are 5.56% and 4.50% better, respectively.

Table 18 shows that the current image-based classification methods have reached 99.5% recall in concrete surface crack defects classification. However, this study aimed to determine the semantic information of each point separately, while the classification methods are not suitable for this purpose. Moreover, the proposed method focused on multiclass point cloud semantic segmentation while the image-based methods focused on binary classification or binary semantic segmentation. To this end, the performance of NVE-DGCNN recall is higher than previous image-based semantic segmentation methods.

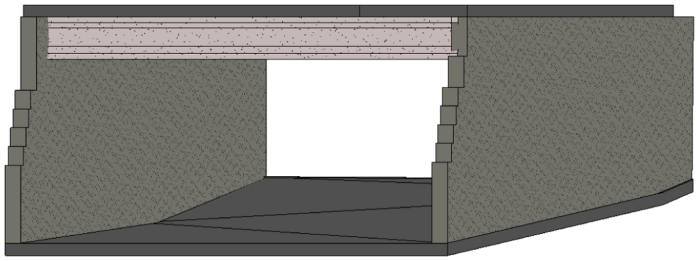
Moreover, the previous point cloud-based works for concrete defects such as [24,33,34,40] utilized different metrics (i.e. error) or visualization approaches to show the results. Therefore, the results are incomparable.

8. Summary, conclusions, and future work

This paper developed a new method for point cloud-based defect semantic segmentation (NVE-DGCNN) to automate the inspection process of concrete surface defects, including cracks and spalls, without transforming the point cloud into other representations. Moreover, this paper investigated two main characteristics related to surface defects (i.e. normal vector and depth). The challenges related to the size of the dataset and imbalanced classes were studied. Sensitivity analysis was applied to capture the best combination of hyperparameters and investigate their effects on the network performance. Furthermore, post-



(a) Imported point cloud data in Revit



(b) 3D model of the bridge

Fig. 11. 3D model of the bridge based on point cloud data.

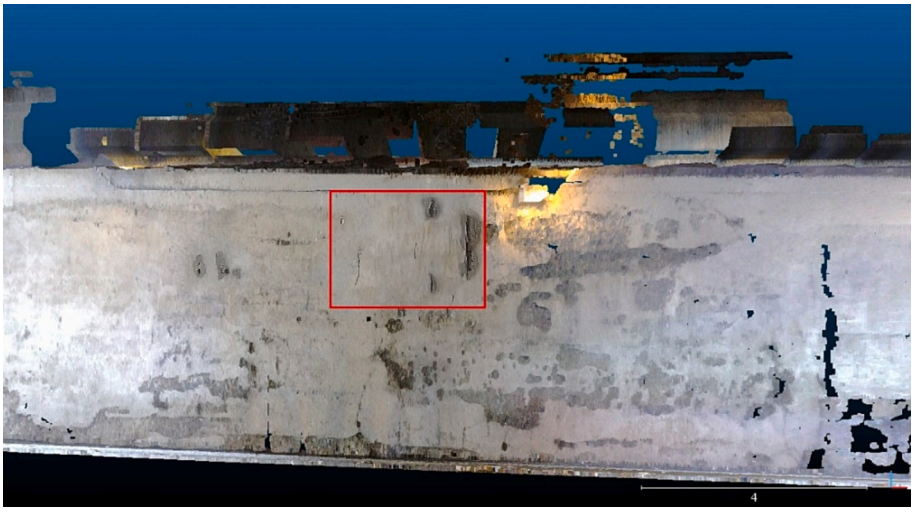


Fig. 12. Selected part from point cloud data.

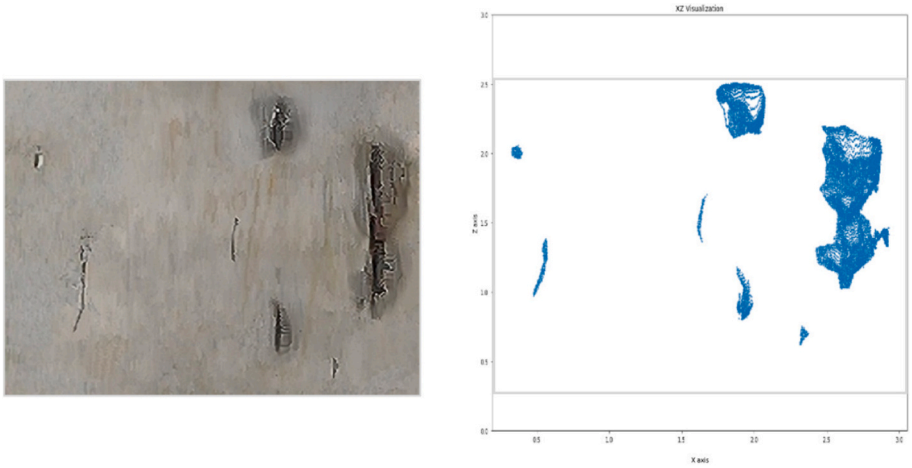
processing of the results of the concrete surface defects semantic segmentation was done to semi-automate the process of as-inspected modeling.

The network's performance was improved by modifying the network (e.g., KNN for EdgeConv and the loss function) and by augmenting the dataset (i.e. by flipping the point cloud data). The testing showed the usefulness and robustness of the proposed method in detecting concrete surface defects from 3D point cloud data. Moreover, the results showed that the normal vector can be an important factor in the learning process of the model and detecting the edge of cracks.

This research results in the following contributions: (1) Developing a method (NVE-DGCNN) for point cloud-based concrete surface defects semantic segmentation; and (2) Developing a semi-automated process for as-inspected modeling.

The following conclusions can be drawn from the above contribution: (1) NVE-DGCNN resulted in 98.56% and 96.50% recalls for semantic segmentation of cracks and spalls, respectively. NVE-DGCNN is more accurate than other point cloud-based methods; (2) The sensitivity analysis results showed that decreasing the size of blocks to less than 30 $\times$ 30 *cm* decreased the recall, as increasing the density of blocks can cause overfitting or failure in Euclidean distance computing. Moreover, decreasing the stride to 25% improved the network performance in terms of recall for the block size of 40 $\times$ 40 *cm*. However, decreasing the stride to 50% was not beneficial and decreased the recall. Finally, the sensitivity analysis showed that NVE-DGCNN is not very sensitive to the points density; (3) The case study showed that deeper cracks and spalls in the dataset are easier to detect. In deeper samples, the recalls for cracks and spalls reached 99.38% and 99.41%, respectively; and (4) The semi-automated process of as-inspected modeling made it possible to manage and visualize the detected defects by collecting their dimensions and identifying the conditions on the 3D model.

Despite the above-mentioned contributions, this paper has some limitations that should be addressed in the future. The limitations can be organized in two categories as follows:

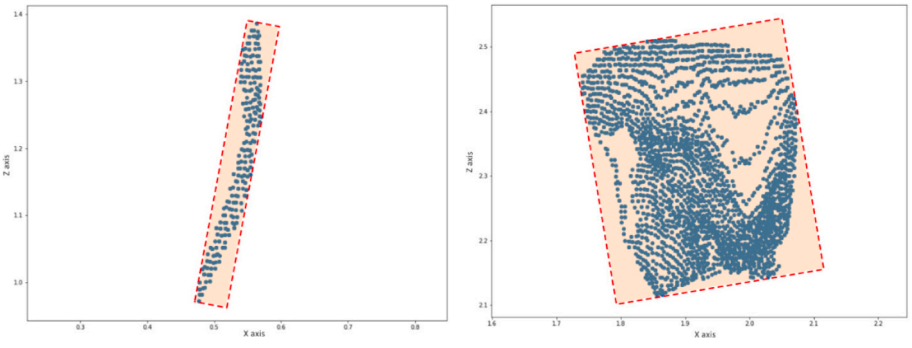


(a) Original point cloud (b) Detected defects using NVE-DGCNN

Fig. 13. Visualization of detected cracks and spalls of the sample.

(a) Clusters of cracks (b) Clusters of spalls

Fig. 14. Clusters in the sample.



(a) Cluster Number 0 of cracks (b) Cluster Number 2 of spalls

Fig. 15. An example of MBBs for cracks and spalls clusters in the sample.

Table 15
Geometric and semantic information of crack defects in the sample.

Cluster number	Length (*mm*)	Width (*mm*)	Depth (*mm*)	Severity	Condition
0	423	45	4	Wide	Poor
1	339	41	4	Wide	condition

(1) Limitations related to the point cloud-based semantic segmentation

- The implementation of the model for point cloud semantic segmentation may entail practical challenges, such as significant time consumption and computational resource requirements.
- A larger dataset is expected to improve the learning process resulting in better performance of the model. Therefore, future work will focus on collecting and preparing more data to enlarge the dataset.

Table 16
Geometric and semantic information of spall defects in the sample.

Cluster number	Length (*mm*)	Width (*mm*)	Depth (*mm*)	Severity	Condition
0	90	69	8	Light	
1	376	108	35	Medium	
2	391	324	68	Medium	Poor condition
3	127	53	3	Light	
4	1168	495	64	Very sever	

Moreover, due to computing resource limitations (i.e. memory and processors limitation), it was impossible to study the effect of increasing the number of input points of the model to more than 12,288 per block. Having more computing resources will make the opportunity to expand the sensitivity analysis.

- The proposed method only considered flat surfaces. The performance of the network on curved surfaces needs more investigation.
- In the future, the dataset can be classified into more classes in order to consider the levels of severity as described in OSIM.
- The effects of each point feature on the performance of the NVE-DGCNN network were not examined independently (e.g. color). Therefore, in the future, it is important to investigate the impact of each feature on network performance.
- The resolution of the point cloud data utilized in the study poses a limitation on the achievable accuracy and level of detail in the segmentation results. Lower point cloud resolutions may lead to reduced precision and potential loss of important semantic information, thereby impacting the overall effectiveness of the segmentation model.
- The general applicability of the method in other civil infrastructures will be tested in our future work. The applicability of the developed method in other civil infrastructures may be subject to constraints and variations inherent to different types of structures. Factors such as varying geometric complexities, and material properties can influence the performance and generalizability of the segmentation method, necessitating careful consideration and adaptation when it is applied to different infrastructure scenarios.

(2) Limitations related to the as-inspected modeling

- As-inspected modeling will help store the inspection results efficiently and precisely, resulting in tracking and analysis of the defect changes throughout the lifecycle. The version control (i.e. time-series) and tracking of the changes of the as-inspected BIM models will be investigated in the future.
- In this research, the defect's semantic information, such as the severity level, is added manually to the BIM model. Future work will

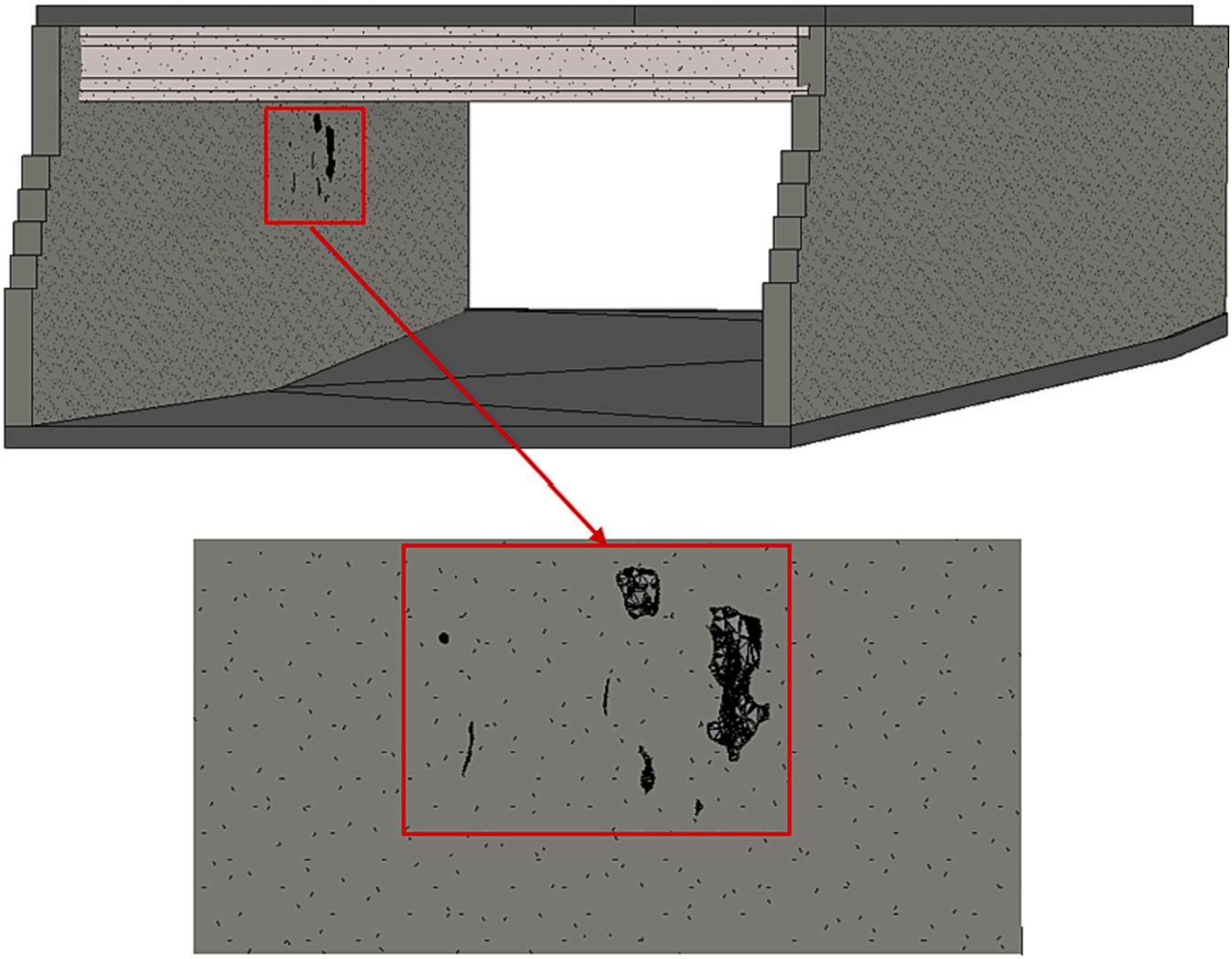


Fig. 16. Imported bridge abutment's defect objects in Revit.

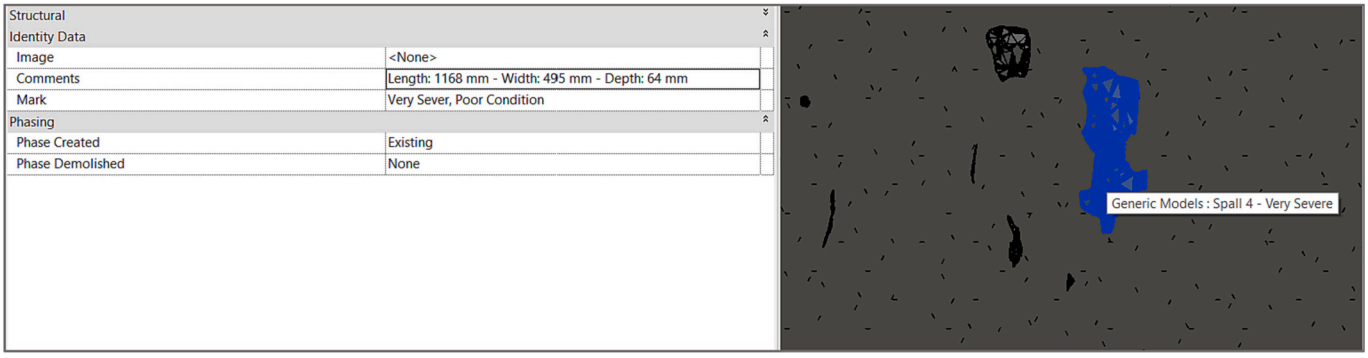


Fig. 17. An example of semantic information for cluster Number 4 of spalls in BIM.

Table 17
Comparison between best results of the NVE-DGCNN and SNEPointNet++ (%).

Method	Cracks				Spalls				No defect			
	Precision	Recall	F1 score	IOU	Precision	Recall	F1 score	IOU	Precision	Recall	F1 score	IOU
SNEPointNet++ [86]	73.3	93.0	81.9	69.2	89.9	92.0	90.6	82.5	99.3	98.8	99.0	98.1
NVE-DGCNN	95.98	98.56	97.25	94.65	95.55	96.50	96.02	92.35	99.46	99.27	99.36	98.73

Table 18
Comparison between NVE-DGCNN and image-based computer vision methods.

Author	Year	Image classification	Image semantic segmentation	Point cloud semantic segmentation	Surface material	CCN input data	Type of defects	Precision (%)	Recall (%)	F1-Score (%)
Yang et al. [87]	2018	–	✓	–	Concrete wall, pavement	Image	Cracks	81.73	78.97	79.95
Liu et al. [88]	2019	–	✓	–	Concrete	Image	Cracks	90	91	90
Lopez Droguett et al. [89]	2020	–	✓	–	Concrete	Image	Cracks	N/A	97.9	97.5
Ali et al. [90]	2021	✓	–	–	Concrete	Image	Cracks	99.7	85	91.8
Le et al. [91]	2021	✓	–	–	Concrete	Image	Cracks	96.5	98.8	97.7
Vignesh et al. [92]	2021	✓	–	–	Concrete	Image	Cracks	96.69	99.55	98.1
Mohammed Abdelkader et al. [93]	2021	–	✓	–	Concrete	Image	Spalls	N/A	N/A	90.98
NVE-DGCNN	ours	–	–	✓	Concrete	Point cloud	Cracks Spalls	95.98 95.55	98.56 96.50	97.25 96.02

focus on a fully automated approach to integrate the defect's semantic information with as-inspected modeling.

CRediT authorship contribution statement

Fardin Bahreini: Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Writing – original draft. **Amin Hammad:** Conceptualization, Funding acquisition, Methodology, Project administration, Resources, Supervision, Writing – review & editing.

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

The authors are unable or have chosen not to specify which data has been used.

References

[1] C. Balaguer, A. Gimenez, C.M. Abderrahim, ROMA robots for inspection of steel based infrastructures, Ind. Robot. 29 (3) (2002) 246–251, https://doi.org/10.1108/01439910210425540.
[2] Y. Bao, Z. Tang, H. Li, Y. Zhang, Computer vision and deep learning–based data anomaly detection method for structural health monitoring, Struct. Health Monit. 18 (2) (2019) 401–421, https://doi.org/10.1177/1475921718757405.
[3] B. Guldur, Y. Yan, J.F. Hajjar, Condition assessment of bridges using terrestrial laser scanners, in: *Proceedings of Structures Congress on Bridges and Other Structures*, Portland, Oregon, 2015, https://doi.org/10.1061/9780784479117.031.
[4] M.R. Jahanshahi, J.S. Kelly, S.F. Masri, G.S. Sukhatme, A survey and evaluation of promising approaches for automatic image-based defect detection of bridge structures, Struct. Infrastruct. Eng. 5 (6) (2009) 455–486, https://doi.org/10.1080/15732470801945930.
[5] C. Koch, I. Brilakis, Pothole detection in asphalt pavement images, Adv. Eng. Inform. 25 (3) (2011) 507–515, https://doi.org/10.1016/j.aei.2011.01.002.
[6] T.C. Hutchinson, Z. Chen, Improved image analysis for evaluating concrete damage, J. Comput. Civ. Eng. 20 (3) (2006) 210–216, https://doi.org/10.1061/(ASCE)0887-3801(2006)20:3(210).
[7] L. Barazzetti, M. Scaioni, Crack measurement: Development, testing and applications of an automatic image-based algorithm, SPRS J. Photogramm. Remote Sens. 64 (3) (2009) 285–296, https://doi.org/10.1016/j.isprsjprs.2009.02.004.
[8] U. Hampel, H.G. Maas, Cascaded image analysis for dynamic crack detection in material testing, ISPRS J. Photogramm. Remote Sens. 64 (9) (2009) 345–350, https://doi.org/10.1016/j.isprsjprs.2008.12.006.
[9] R.S. Adhikari, O. Moselhi, A. Bagchi, Image-based retrieval of concrete crack properties, Autom. Constr. 1 (34) (2012) 180–194, https://doi.org/10.1016/j.autcon.2013.06.011.

[10] Y. Huang, B. Xu, Automatic inspection of pavement cracking distress, J. Electron. Imaging 5 (1) (2006) 013017, https://doi.org/10.1117/1.2177650.
[11] Y. Sun, E. Salari, E. Chou, Automated pavement distress detection using advanced image processing techniques, in: *Proceedings of IEEE International Conference on Electro/Information Technology*, Windsor, Ontario, Canada, 2009, https://doi.org/10.1109/EIT.2009.5189645.
[12] C. Koch, K. Georgieva, V. Kasireddy, B. Akinci, P. Fieguth, A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure, Adv. Eng. Inform. 29 (2) (2015) 196–210, https://doi.org/10.1016/j.aei.2015.01.008.
[13] C. Haas, Evolution of an automated crack sealer: a study in construction technology development, Autom. Constr. 4 (4) (1996) 293–305, https://doi.org/10.1016/0926-5805(95)00010-0.
[14] C.J. Smith, K.K. Adendorff, Advantages and limitations of an automated visual inspection system, South Africa J. Indus. Eng. 5 (1) (1991) 27–36, https://doi.org/10.7166/5-1-423.
[15] D.F. Laefer, L. Truong-Hong, H. Carr, M. Singh, Crack detection limits in unit based masonry with terrestrial laser scanning, Ndt & E Int. 62 (1) (2014) 66–76, https://doi.org/10.1016/j.ndteint.2013.11.001.
[16] D.H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, Pattern Recogn. 13 (2) (1981) 111–122, https://doi.org/10.1016/0031-3203(81)90009-1.
[17] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) (1981) 381–395, https://doi.org/10.1145/358669.358692.
[18] E. Grilli, F. Menna, F. Remondino, A review of point clouds segmentation and classification algorithms, in: *Proceedings of the ISPRS International Archives on the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Wuhan, China, 2017, https://doi.org/10.1145/358669.358692.
[19] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, J.M. Solomon, Dynamic graph cnn for learning on point clouds, ACM Trans. Graph. 38 (5) (2019) 1–12, https://doi.org/10.1145/3326362.
[20] W. Liu, J. Sun, W. Li, T. Hu, P. Wang, Deep learning on point clouds and its application: A survey, Sensors 19 (9) (2019), https://doi.org/10.3390/s19194188, 4188/1-22.
[21] V. Kasireddy, B. Akinci, Towards the integration of inspection data with bridge information models to support visual condition assessment, in: *Proceedings of the ASCE International Workshop on Computing in Civil Engineering*, Austin, Texas, 2015, https://doi.org/10.1061/9780784479247.080.
[22] F. Bahreini, A. Hammad, Point cloud semantic segmentation of concrete surface defects using dynamic graph CNN, in: Proceedings of the 38th International Symposium on Automation and Robotics in Construction (ISARC), 2021, pp. 379–386, https://doi.org/10.22260/ISARC2021/0053.
[23] M. Rashidi, M. Mohammadi, S. Sadeghlou Kivi, M.M. Abdolvand, L. Truong-Hong, B. Samali, A decade of modern bridge monitoring using terrestrial laser scanning: review and future directions, Remote Sens. 12 (22) (2020), https://doi.org/10.3390/rs12223796, 3796/1-34.
[24] L. Truong-Hong, H. Falter, D. Lennon, D.F. Laefer, Framework for bridge inspection with laser scanning, in: *Proceedings of the EASEC-14 Conference on Structural Engineering and Construction*, Minh City, Vietnam, 2016. http://hdl.handle.net/10197/7476.
[25] P. Tang, B. Akinci, D. Huber, Quantification of edge loss of laser scanned data at spatial discontinuities, Autom. Constr. 18 (8) (2009) 1070–1083, https://doi.org/10.1016/j.autcon.2009.07.001.
[26] G. Teza, A. Galgaro, F. Moro, Contactless recognition of concrete surface damage from laser scanning and curvature computation, NDT & E Int. 42 (4) (2009) 240–249, https://doi.org/10.1016/j.ndteint.2008.10.009.
[27] M.J. Olsen, F. Kuester, B.J. Chang, T.C. Hutchinson, Terrestrial laser scanning-based structural damage assessment, J. Comput. Civ. Eng. 24 (3) (2010) 264–272, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000028.
[28] E.B. Anil, B. Akinci, J.H. Garrett, O. Kurc, Characterization of laser scanners for detecting cracks for post-earthquake damage inspection, in: *Proceedings of the International Symposium on Automation and Robotics in Construction*, Montreal, Canada, 2013, https://doi.org/10.22260/ISARC2013/0034.
[29] X. Xu, H. Yang, Intelligent crack extraction and analysis for tunnel structures with terrestrial laser scanning measurement, Adv. Mech. Eng. 11 (9) (2019) 1–7, https://doi.org/10.1177/1687814019872650.
[30] M. Makuch, P. Gawronek, 3D point cloud analysis for damage detection on hyperboloid cooling tower shells, Remote Sens. 12 (10) (2020), https://doi.org/10.3390/rs12101542, 1542/1-23.
[31] W. Liu, S. Chen, E. Hauser, LiDAR-based bridge structure defect detection, Exp. Tech. 35 (6) (2011) 27–34, https://doi.org/10.1111/j.1747-1567.2010.00644.x.
[32] J. Armesto-González, B. Riveiro-Rodríguez, D. González-Aguilera, M.T. Rivas-Brea, Terrestrial laser scanning intensity data applied to damage detection for historical buildings, J. Archaeol. Sci. 73 (12) (2010) 3037–3047, https://doi.org/10.1111/j.1747-1567.2010.00644.x.
[33] J. Valença, I. Puente, E. Júlio, H. González-Jorge, P. Arias-Sánchez, Assessment of cracks on concrete bridges using image processing supported by laser scanning survey, Constr. Build. Mater. 146 (1) (2017) 668–678, https://doi.org/10.1016/j.conbuildmat.2017.04.096.
[34] M.K. Kim, H. Sohn, C.C. Chang, Localization and quantification of concrete spalling defects using terrestrial laser scanning, J. Comput. Civ. Eng. 29 (6) (2015) 04014086, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000415.
[35] Y.C.J. Tsai, F. Li, Critical assessment of detecting asphalt pavement cracks under different lighting and low intensity contrast conditions using emerging 3D laser technology, J. Transp. Eng. 138 (5) (2012) 649–656, https://doi.org/10.1061/(ASCE)TE.1943-5436.0000353.
[36] M. Cabaleiro, R. Lindenbergh, W.F. Gard, P. Arias, J.W.G. Van de Kuilen, Algorithm for automatic detection and analysis of cracks in timber beams from LiDAR data, Constr. Build. Mater. 130 (1) (2017) 41–53, https://doi.org/10.1016/j.conbuildmat.2016.11.032.
[37] T. Mizoguchi, Y. Koda, I. Iwaki, H. Wakabayashi, Y. Kobayashi, K. Shirai, Y. Hara, H.S. Lee, Quantitative scaling evaluation of concrete structures based on terrestrial laser scanning, Autom. Constr. 35 (1) (2013) 263–274, https://doi.org/10.1016/j.autcon.2013.05.022.
[38] M. Nasrollahi, N. Bolourian, A. Hammad, Concrete surface defect detection using deep neural network based on lidar scanning, in: *Proceedings of the CSCE Annual Conference on AI and machine learning*, Montreal, Canada, 2019. https://www.csce.ca/elf/apps/CONFERENCEVIEWER/conferences/2019/pdfs/PaperPDFversion_32_0409032519.pdf.
[39] G. Te, W. Hu, A. Zheng, Z. Guo, Rgcnn: Regularized graph cnn for point cloud segmentation, in: *Proceedings of the 26th ACM International Conference on Multimedia*, New York, NY, USA, 2018, https://doi.org/10.1145/3240508.3240621.
[40] B.G. Erkal, J.F. Hajjar, Laser-based surface damage detection and quantification using predicted surface properties, Autom. Constr. 83 (1) (2017) 285–302, https://doi.org/10.1016/j.autcon.2017.08.004.
[41] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, M. Bennamoun, Deep learning for 3d point clouds: a survey, IEEE Trans. Pattern Anal. Mach. Intell. 43 (12) (2020) 4338–4364, https://doi.org/10.1109/TPAMI.2020.3005434.
[42] T. Su, H. Li, S. Zhang, Y. Li, Image segmentation using mean shift for extracting croplands from high-resolution remote sensing imagery, Remote Sens. Lett. 6 (12) (2015) 952–961, https://doi.org/10.1080/2150704X.2015.1093188.
[43] A. Brock, T. Lim, J.M. Ritchie, N. Weston, Generative and discriminative voxel modeling with convolutional neural networks, in *arXiv preprint arXiv:1608.04236*, 2016, https://doi.org/10.48550/arXiv.1608.04236.
[44] C.R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, L.J. Guibas, Volumetric and multi-view cnns for object classification on 3d data, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, NV, USA, 2016, https://doi.org/10.1109/CVPR.2016.609.
[45] C.R. Qi, H. Su, K. Mo, L.J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Honolulu, HI, USA, 2017, https://doi.org/10.48550/arXiv.1612.00593.
[46] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3D shapenets: A deep representation for volumetric shapes, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Boston, MA, USA, 2015, https://doi.org/10.48550/arXiv.1406.5670.
[47] I. Armeni, O. Sener, A.R. Zamir, H. Jiang, I. Brilakis, M. Fischer, S. Savarese, 3D semantic parsing of large-scale indoor spaces, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, https://doi.org/10.1109/CVPR.2016.170.
[48] R. Pierdicca, M. Paolanti, F. Matrone, M. Martini, C. Morbidoni, E.S. Malinverni, E. Frontoni, A.M. Lingua, Point cloud semantic segmentation using a deep learning framework for cultural heritage, Remote Sens. 12 (6) (2020), https://doi.org/10.3390/rs12061005, 1005/1-23.
[49] J.M. Davila Delgado, N. De Battista, I. Brilakis, C. Middleton, Design and data modelling of fibre optic systems to monitor reinforced concrete structural elements, in: *Proceedings of Construction Research Congress on Old and New Construction Technologies*, San Juan, Puerto Rico, 2016, https://doi.org/10.1061/9780784479827.228.
[50] J. Chen, T. Bulbul, J.E. Taylor, G. Olgun, A case study of embedding real-time infrastructure sensor data to BIM, in: *Proceedings of Construction Research Congress on Construction in a Global Network*, Atlanta, Georgia, 2014, https://doi.org/10.1061/9780784413517.028.
[51] A. Hammad, F. Vahdatikhaki, A. Albahri, S. Salimi, D. Moghaddam, A. Jahromi, Framework for life-cycle infrastructure information modeling and management, in: *Proceedings of Annual Conference of the Canadian Society for Civil Engineering on Construction*, Montreal, Canada, 2013 https://csce.ca/elf/apps/CONFERENCEVIEWER/conferences/2013/pdfs/construction/160.pdf.
[52] G. Mailhot, S. Busuioc, Application of long range 3D laser scanning for remote sensing and monitoring of complex bridge structures, in: *Proceedings of the 7th International Conference of Short & Medium*, Montreal, Canada, 2006 https://www.proceedings.com/01705.html.
[53] A. Hammad, C. Zhang, Y. Hu, E. Mozaffari, Mobile model-based bridge lifecycle management system, Comput. Aid. Civil Infrastruct. Eng. 21 (7) (2006) 530–547, https://doi.org/10.1111/j.1467-8667.2006.00456.x.
[54] H. Hamledari, E. Rezazadeh Azar, B. McCabe, IFC-based development of as-built and as-is BIMs using construction and facility inspection data: Site-to-BIM data transfer automation, J. Comput. Civ. Eng. 32 (2) (2018), https://doi.org/10.1061/(ASCE)CP.1943-5487.0000727, 04017075/1-15.
[55] L. Ma, R. Sacks, R. Zeibak-Shini, Information modeling of earthquake-damaged reinforced concrete structures, Adv. Eng. Inform. 29 (3) (2015) 396–407, https://doi.org/10.1016/j.aei.2015.01.007.
[56] F. Tanaka, M. Hori, M. Onosato, H. Date, S. Kanai, Bridge information model based on IFC standards and web content providing system for supporting an inspection process, in: *Proceedings of the 16th International Conference on Computing in Civil and Building Engineering*, Osaka, Japan, 2016 http://www.see.eng.osaka-u.ac.jp/seeit/icccbe2016/Proceedings/Full_Papers/144-301.pdf.
[57] F. Tanaka, M. Tsuchida, M. Onosato, H. Date, S. Kanai, Y. Hada, M. Nakao, H. Kobayashi, E. Hasegawa, T. Sugawara, T. Oyama, Bridge Information Modeling based on IFC for supporting maintenance management of existing bridges, in: *Proceedings of the 17th International Conference on Computing in Civil and Building*

Engineering, Tampere, Finland, 2018 https://icccbe2018.exordo.com/files/papers/149/final_draft/Final_ICCCBE2018_tanaka.pdf.
[58] R. Sacks, A. Kedar, A. Borrmann, L. Ma, I. Brilakis, P. Hüthwohl, S. Daum, U. Kattel, R. Yosef, T. Liebich, B.E. Barutcu, SeeBridge as next generation bridge inspection: overview, information delivery manual and model view definition, Autom. Constr. 90 (1) (2018) 134–145, https://doi.org/10.1016/j.autcon.2018.02.033.
[59] M. Artus, C. Koch, State of the art in damage information modeling for RC bridges–a literature review, Adv. Eng. Inform. 46 (1) (2020), https://doi.org/10.1016/j.aei.2020.101171, 101171/1-16.
[60] P. Hüthwohl, I. Brilakis, A. Borrmann, R. Sacks, Integrating RC bridge defect information into BIM model, Am. Soc. Civil Eng. 32 (3) (2018), https://doi.org/10.1061/(ASCE)CP.1943-5487.0000744, 04018013/1-14.
[61] A. Hamdan, R.J. Scherer, A generic model for the digitalization of structural damage, in: *Proceedings of the Sixth International Symposium on Life-Cycle Civil Engineering*, Ghent, Belgium, 2018, https://doi.org/10.1201/9781315228914.
[62] A.H. Hamdan, J. Taraben, M. Helmrich, T. Mansperger, G. Morgenthal, R. J. Scherer, A semantic modeling approach for the automated detection and interpretation of structural damage, Autom. Constr. 128 (1) (2021), https://doi.org/10.1016/j.autcon.2021.103739, 103739/1-17.
[63] M. Artus, C. Koch, Modeling geometry and semantics of physical damages using IFC, in: *EG-ICE 2020 Workshop on Intelligent Computing in Engineering*, Universitätsverlag der TU Berlin, 2020, https://doi.org/10.14279/depositonce-9977.
[64] M. Artus, M.S.H. Alabassy, C. Koch, A BIM based framework for damage segmentation, modeling, and visualization using IFC, Appl. Sci. 12 (6) (2021), https://doi.org/10.3390/app12062772.
[65] D. Isailović, V. Stojanovic, M. Trapp, R. Richter, R. Hajdin, J. Döllner, Bridge damage: Detection, IFC-based semantic enrichment and visualization, Autom. Constr. 112 (1) (2020), https://doi.org/10.1016/j.autcon.2020.103088, 103088/1-22.
[66] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep Learning, MIT press, Cambridge, MA, USA, 2016 https://mitpress.mit.edu/9780262035613/deep-learning/.
[67] S. Jadon, A survey of loss functions for semantic segmentation, in: *PROCEEDINGS OF IEEE CONFERENCE ON COMPUTATIONAL INTELLIGENCE IN BIOINFORMATICS AND COMPUTATIONAL BIOLOGY*, Santiago, Chile, 2020, https://doi.org/10.1109/CIBCB48159.2020.9277638.
[68] J. Hyeon, W. Lee, J.H. Kim, N. Doh, NormNet: Point-wise normal estimation network for three-dimensional point cloud data, Int. J. Adv. Robot. Syst. 16 (4) (2019), https://doi.org/10.1177/1729881419857532, 1729881419857532/1-11.
[69] HDF-Group, HDF5 User's Guide, 2018. [Online]. Available: https://portal.hdfgroup.org/display/HDF5/HDF5+User%27s+Guide [Accessed May 2021].
[70] Q. Liu, M. Deng, Y. Shi, J. Wang, A density-based spatial clustering algorithm considering both spatial proximity and attribute similarity, Comput. Geosci. 46 (1) (2012) 296–309, https://doi.org/10.1016/j.cageo.2011.12.017.
[71] K.N. Ahmed, T.A. Razak, A comparative study of different density based spatial clustering algorithms, Int. J. Comput. Appl. 99 (8) (2014) 18–25, https://doi.org/10.5120/17393-7942.
[72] D. Sidlauskas, S. Chester, E.T. Zacharatou, A. Ailamaki, Improving spatial data processing by clipping minimum bounding boxes, in: *Proceedings of IEEE 34th International Conference on Data Engineering*, Paris, France, 2018, https://doi.org/10.1109/ICDE.2018.00046.
[73] G. Barequet, S. Har-Peled, Efficiently approximating the minimum-volume bounding box of a point set in three dimensions, J. Algorithms 38 (1) (2001) 91–109, https://doi.org/10.1006/jagm.2000.1127.
[74] Ontario Ministry of Transportation Ontario, Ontario Structure Inspection Manual (OSIM), Ontario ministry of transportation, Provincial highways management division, Highway standards branch, Bridge office, Ontario, Canada, 2018. Available :https://www.library.mto.gov.on.ca/SydneyPLUS/Sydney/Portal/default.aspx?component=AAAAIY&record=2cc7e50c-3d41-4468-90f1-0788368ce945&lang=en-US ISBN 9781486815081.
[75] NICP algorithm, Pyoints Documentation [Online]. Available: https://laempy.github.io/pyoints/tutorials/icp.html#References [Accessed May 2022].
[76] J. Serafin, G. Grisetti, NICP: Dense normal based point cloud registration, in: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Kyoto, Japan, 2015, https://doi.org/10.1109/IROS.2015.7353455.
[77] Leica Geosystems, 3DReshaper Point Cloud Processing Software, May 2021 [Online]. Available: https://leica-geosystems.com/products/laser-scanners/software/3dreshaper.
[78] M. Misol Monzo, Dynamo Mesh Toolkit, May 2020 [Online]. Available: https://github.com/DynamoDS/Dynamo/wiki/Dynamo-Mesh-Toolkit [Accessed June 2021].
[79] FARO Technologies Inc., Faro Laser Scanner Focus 3D X 130 -NEO-Tech, 2010 [Online]. Available: http://www.neo-tech.co.uk/assets/faro-focus-3d-x-130-laser-scanner-brochure.pdf [Accessed 5 JUN 2020].
[80] D. Girardeau-Montaut, M. Roux, R. Marc, G. Thibault, Change detection on points cloud data acquired with a ground laser scanner, in: *Proceedings of the International Archives on Photogrammetry*, Remote Sensing and Spatial Information Sciences, Hannover, Germany, 2005 https://www.danielgm.net/phd/isprs_laserscanning_2005_dgm.pdf.
[81] B. Lohani, R. Singh, Effect of data density, scan angle, and flying height on the accuracy of building extraction using LiDAR data, Geocarto Int. 32 (2) (2008) 81–94, https://doi.org/10.1080/10106040701207100.
[82] FARO Technologies Inc., Quality Setting Function on the Focus3D, 2019 [Online]. Available: https://knowledge.faro.com/Hardware/3D_Scanners/Focus/Quality_Setting_Function_on_the_Focus3D [Accessed 5 JUN 2020].
[83] FARO Technologies Inc., User Manual FARO Laser Scanner Focus 3D, 2011 [Online]. Available: https://downloads.faro.com/index.php/s/89t7CjYKcjGZd3R [Accessed 5 JUN 2020].
[84] J. Brownlee, Overfitting and underfitting with machine learning algorithms, Machine Learning Mastery, 2016 [Online]. Available: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/ [Accessed 2021 May 03].
[85] X. Wang, R. Girshick, A. Gupta, K. He, Non-local neural networks, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Salt Lake City, UT, USA, 2018, https://doi.org/10.1109/CVPR.2018.00813.
[86] N. Bolourian, M. Nasrollahi, F. Bahreini, A. Hammad, Point-based concrete surface defect semantic segmentation, J. Comput. Civ. Eng. 37 (2) (2023) 04022056, https://doi.org/10.1061/JCCEE5.CPENG-5009.
[87] X. Yang, H. Li, Y. Yu, X. Luo, T. Huang, X. Yang, Automatic pixel-level crack detection and measurement using fully convolutional network, Comput. Aid. Civil Infrastruct. Eng. 33 (12) (2018) 1090–1109, https://doi.org/10.1111/mice.12412.
[88] Z. Liu, Y. Cao, Y. Wang, W. Wang, Computer vision-based concrete crack detection using U-net fully convolutional networks, Autom. Constr. 104 (1) (2019) 129–139, https://doi.org/10.1016/j.autcon.2019.04.005.
[89] E. Lopez Droguett, J. Tapia, C. Yanez, R. Boroschek, Semantic segmentation model for crack images from concrete bridges for mobile devices, Proc. Instit. Mech. Eng. Part O: J. Risk Reliability 236 (4) (2020) 570–583, https://doi.org/10.1177/1748006X20965111.
[90] L. Ali, F. Alnajjar, H.A. Jassmi, M. Gochoo, W. Khan, M.A. Serhani, Performance evaluation of deep CNN-Based crack detection and localization techniques for concrete structures, Sensors 21 (5) (2021), https://doi.org/10.3390/s21051688, 1688/1-22.
[91] T.T. Le, V.H. Nguyen, M.V. Le, Development of deep learning model for the recognition of cracks on concrete surfaces, Appl. Comput. Intell. Soft Comput. 2021 (1) (2021) 1687–9724, https://doi.org/10.1155/2021/8858545.
[92] R. Vignesh, B. Narenthiran, S. Manivannan, R.A. Murugan, V. RajKumar, Concrete bridge crack detection using convolutional neural network, in: *Proceedings of the International Conference Materials, Design, and Manufacturing for Sustainable Environment*, Springer, Singapore, 2021, https://doi.org/10.1007/978-981-15-9809-8_58.
[93] E. Mohammed Abdelkader, O. Moselhi, M. Marzouk, T. Zayed, Entropy-based automated method for detection and assessment of spalling severities in reinforced concrete bridges, J. Perform. Constr. Facil. 35 (1) (2021), https://doi.org/10.1061/(ASCE)CF.1943-5509.0001544, 04020132/1-25.