Point Cloud–Based Concrete Surface Defect Semantic Segmentation

Neshat Bolourian, Ph.D.[1]; Majid Nasrollahi[2]; Fardin Bahreini[3]; and Amin Hammad[4]

Abstract: Visual inspection is one of the main approaches for annual bridge inspection. Light detection and ranging (LiDAR) scanning is a new technology, which is beneficial because it collects the point clouds and the third dimension of the scanned objects. Deep learning (DL)-based methods have attracted researchers' attention for concrete surface defect detection. However, no point cloud–based DL method currently is available for semantic segmentation of bridge surface defects without converting the data set into other representations, which results in increasing the size of the data set. Moreover, most of the current point cloud–based concrete surface defect detection methods focus on only one type of defect. On the other hand, a data set plays a key role in DL. Therefore, the lack of publicly available point cloud data sets for bridge surface defects is one of the reasons for the lack of studies in this area. To address these issues, this paper created a publicly available point cloud data set for concrete bridge surface defect detection, and developed a point cloud–based semantic segmentation DL method to detect different types of concrete surface defects. Surface Normal Enhanced PointNet++ (SNEPointNet++) was developed for semantic segmentation of concrete bridge surface defects (i.e., cracks and spalls). SNEPointNet++ focuses on two main characteristics related to surface defects (i.e., normal vector and depth) and considers the issues related to the data set (i.e., imbalanced data set). The data set, which was collected from four concrete bridges and classified into three classes (cracks, spalls, and no defect), is made available for other researchers. The model was trained and evaluated using 60% and 20% of the data set, respectively. Testing on the remaining part of the data set resulted in 93% and 92% recall for cracks and spalls, respectively. Spalls of the segments deeper than 7 cm (severe spalls) can be detected with 99% recall. **DOI: 10.1061/JCCEE5.CPENG-5009.** © *2022 American Society of Civil Engineers.*

Author keywords: Deep learning (DL); Bridge inspection; Light detection and ranging (LiDAR); Point cloud; Surface defect detection; Semantic segmentation.

Introduction

In Canada, most bridges have exceeded more than half of their expected service life, leaving a heavy burden of investment for rehabilitation and maintenance (Gagnon et al. 2008). Despite the development and use of several new techniques for detecting bridge defects, visual inspection remains the main approach in detecting surface defects, such as cracks and spalling (Laefer et al. 2014). Therefore, many researchers have explored new methods for increasing visual inspection accuracy, efficiency, and safety. The main problem of visual inspection is its reliance for the most part on manual data collection using the naked eye, which is subjective and time-consuming (Dorafshan and Maguire 2018). Several technologies [i.e., cameras, and three-dimensional (3D) light detection and ranging (LiDAR)] recently have been used to detect concrete surface defects (e.g., cracks) automatically and accurately.

Deep learning (DL)-based methods have attracted researchers' attention for concrete surface defect detection (Mohammed Abdelkader et al. 2021; Wang et al. 2022). In computer vision, DL-based methods can be used for applications such as classification, semantic segmentation, and instance segmentation. In terms of detecting the defects, classification models are able to identify the existence of a defect in the input, whereas semantic segmentation refers to labeling each point individually (Qi et al. 2017a). Instance segmentation provides an even more detailed analysis by distinguishing between different instances with the same label, in addition to labeling each point (Jiang et al. 2020). Compared with classification, semantic segmentation, which can be the initial step for instance segmentation, is more beneficial for surface defect detection because it provides detailed information about each point.

In DL, a data set plays a key role in terms of variety, diversity, accuracy, and size. In other words, a large high-quality data set leads to better learning and can help to increase the performance and generalization of the trained model. Many image-based DL methods in this area have been applied on publicly available data sets [i.e., MS-COCO (Lee et al. 2019) and SDNET2018 (Maguire et al. 2018)]. The lack of publicly available point cloud data sets for bridge surface defects is one of the reasons for the lack of studies in the area of point cloud–based methods.

The paper is organized as follows. First, the related studies are reviewed and the objectives are set based on the gaps in previous studies. Then, all the aspects which are considered in the proposed method and the framework of adjusting PointNet++ are elaborated. This is followed by the implementation and case study section.

[1]Research Assistant, Dept. of Building, Civil and Environmental Engineering, Concordia Univ., 1515 Sainte-Catherine St. West, Montreal, QC, Canada H3G 2W1. Email: n_bolour@encs.concordia.ca

[2]Graduate Student, Artificial Intelligence and Risk Management Consulting Ltd., 1-1000 Lorimer Blvd., Winnipeg, MB, Canada R3P 1C8. Email: majid.nasrollahi@airmcosulting.com

[3]Ph.D. Candidate, Dept. of Building, Civil and Environmental Engineering, Concordia Univ., 1515 Sainte-Catherine St. West, Montreal, QC, Canada H3G 2W1. ORCID: https://orcid.org/0000-0002-6832-2597. Email: f_bahrei@encs.concordia.ca

[4]Professor, Concordia Institute for Information Systems Engineering, Concordia Univ., 1515 Sainte-Catherine St. West, Montreal, QC, Canada H3G 2W1 (corresponding author). ORCID: https://orcid.org/0000-0002-2507-4976. Email: amin.hammad@concordia.ca

Note. This manuscript was submitted on May 13, 2022; approved on September 13, 2022; published online on November 21, 2022. Discussion period open until April 21, 2023; separate discussions must be submitted for individual papers. This paper is part of the ***Journal of Computing in Civil Engineering***, © ASCE, ISSN 0887-3801.

Finally, the obtained results are discussed and the conclusions of the proposed method are outlined.

Literature Review

Comparison of Camera- and LiDAR-Based Methods

In the visual inspection of concrete bridges, surface defects (e.g., cracks) can be inspected using 3D point cloud and images, which are captured by LiDAR and cameras, respectively. These methods are much more time-efficient than manual measurements (Rashidi et al. 2020). Having the third dimension is the main advantage of point clouds over images. Although images collected by a depth camera can represent the surface depth, several images with different setups must be collected to cover the whole of the structure depth due to the camera's narrow field of view (FoV). Moreover, the registration process of these images is time-consuming (Maru et al. 2021), which further limits their applicability. Another approach to extract the depth of concrete surface defects (i.e., spalling distress) is a regression analysis model based on computer vision and in situ measurements (Dawood et al. 2017). Computer vision using cameras may detect the boundaries of defects more efficiently than LiDAR (Demir and Baltsavias 2012).

The main shortcomings of camera-based methods are the necessity of providing supplementary information (i.e., camera lens and focal length) before analyzing the images, and the high sensitivity level to environmental conditions (e.g., lighting and shading) (Laefer et al. 2014). Although LiDAR-based methods can work properly regardless of any information related to the equipment or target, determining that information may be helpful in improving the inspection method.

Furthermore, most camera-based methods are defined for simple curved or flat concrete surfaces. Consequently, they may fail at analyzing more-complex structures, geometries, and materials (Zhang et al. 2014; Koch et al. 2015). Despite the high initial cost of laser scanners, using them may be more profitable and economical in the long term. Some researchers focused on improving the resolution of image-based reconstructed models (Khaloo et al. 2018). Khaloo et al. (2018) generated a 3D point cloud bridge model using 2,000 high-resolution images, which resulted in 3 times higher local noise and lower precision compared with the scanned bridge model using a LiDAR. Moreover, the quality of the image-based reconstructed model was lower than that of the LiDAR-based model due to the complexity of detecting the interaction of the background and the inspected bridge, especially in the case of low contrast. Therefore, using point clouds is more beneficial than using images for surface defect detection of bridges. Several point cloud–based methods focus only on the point cloud features (x, y, and z), but some other features (i.e., colors and normal vectors) may be useful for detecting surface defects (Kim et al. 2015; Olsen et al. 2010; Teza et al. 2009).

Point Cloud–Based Semantic Segmentation Using DL

DL can be used for semantic segmentation of point clouds. Semantic segmentation aims to separate point clouds into several subsets and label each point according to the corresponding semantic meanings. Point clouds can be used indirectly or directly for semantic segmentation (Guo et al. 2020). Using the two most common indirect methods, multiview-based (Boulch et al. 2018; Yavartanoo et al. 2018; Zhao et al. 2018) and voxel-based methods (Maturana and Scherer 2015; Wang and Lu 2016; Wu et al. 2015), point clouds are transformed into intermediate representations, which are multiview images and voxel grids, respectively. Transforming point clouds into these representations results in voluminous data with unclear invariances in some cases (Grilli et al. 2017). In multiview-based methods, after learning from two-dimensional (2D) projections, the intermediate segmentation results are projected back to the raw point cloud. The generated meshes are complex with combinatorial irregularities (Qi et al. 2017a). Unlike the data set of a voxel-based approach, which contains an ordered grid of point clouds (Pierdicca et al. 2020), a point cloud–based data set does not require specific preprocessing, and the unordered data set is trained directly. Moreover, learning from point clouds is easier than learning from meshes due to their simplicity and unified structure. Due to the higher computational efficiency, point cloud–based methods recently have become popular in the construction industry (Yin et al. 2021).

PointNet (Qi et al. 2017a) is the pioneer point cloud–based DL architecture designed for point cloud classification, part segmentation, and direct semantic segmentation. This network was proposed to overcome the problems related to voxelization and rendering point clouds. Its input is a set of points, which has three main characteristics. First, these points are unordered. Interaction among neighboring points is the second important characteristic of these data sets. The points are not isolated, and the local structure of the combination of neighboring points affects the semantic information of the point sets. The third characteristic of point clouds is being invariant under transformation. These three characteristics were considered in designing the architecture of PointNet. PointNet has two main shortcomings: (1) lack of local context learning, and (2) translation invariance limitation (Yin et al. 2021). To overcome these limitations, PointNet++, proposed by Qi et al. (2017b), has hierarchical feature learning, which uses multiple Vanilla PointNet learners with different scales. Multiscale sampling is very beneficial for defect semantic segmentation, because the sizes of the defects are very different. Moreover, although several point cloud–based networks [i.e., Dynamic Graph Convolutional Neural Network (DGCNN), PointConv, PointCNN, and ShellNet] recently have been developed, PointNet++ is still an efficient and well-established network (Bello et al. 2020). Some studies have focused on the implementation of point cloud–based DL, such as PointNet, PointNet++, DGCNN, and PointCNN, for inspection of sewer defects (Haurum et al. 2021) and underwater pipes (Martin-Abadal et al. 2021). Table 1 presents the recent studies for inspection in the construction industry based on the well-known point cloud–based semantic segmentation models. Based on the literature review, only two studies focused on structural concrete surface defect detection (e.g., cracks and spalls).

Concrete Surface Defect Detection Using Point Cloud

Damage detection using point clouds can be based on geometrical features or/and color information (Mohammadi et al. 2019). Hou et al. (2017) proposed the combination of clustering algorithms (e.g., k-means, fuzzy c-means, subtract, and density-based spatial) and color [red–green–blue (RGB)] and intensity information of the point cloud data set to detect surface defects such as metal corrosion and section losses. The results showed that due to the effect of lighting conditions on color information, this information is less reliable than intensity information (Hou et al. 2017).

Liu et al. (2011) presented an algorithm to detect defects based on distance and gradient criteria of concrete bridge surfaces. This method was applied to a bridge cap to detect and quantify mass losses. Laefer et al. (2014) proposed a method for using terrestrial laser scanning (TLS) to detect cracks in unit-block masonry (i.e., stone, brick, or concrete masonry units), which resulted in overestimating the crack width. Guldur et al. (2015) developed a

Table 1. Use of point cloud–based methods in inspection (construction industry)

Reference	Objective	Classes	Method	Results (%)	Data set
Nasrollahi et al. (2019)	Concrete surface defect detection	Two classes (defect and no defect)	PointNet	OA = 85.7	21.5 million points
Pierdicca et al. (2020)	Semantic segmentation of historical architectural elements	10 classes (arc, column, decoration, floor, door, wall, window, stairs, vault, and roof)	PointNet	OA = 21.6 mIoU = 10.9	1 million points
			PointNet++	OA = 24.5 mIoU = 18.0	
			DGCNN	OA = 54.7 mIoU = 35.8	
			PCNN	OA = 39.5 mIoU = 33.1	
			Modified DGCNN	OA = 71.6 mIoU = 37.7	
Kim and Kim (2020)	Comparison of bridge component classification DL methods	Six classes (abutment, slab, pier, girder, surface, and background)	PointNet	OA = 93.8 mIoU = 84.3	N/A
			PointCNN	OA = 92.6 mIoU = 76.8	
			DGCNN	OA = 94.5 mIoU = 86.9	
Kim et al. (2020)	Bridge components segmentation	Three classes (deck, pier, and background)	PointNet	OA = 94 mIoU = 84	N/A
Haurum et al. (2021)	Built a publicly available sewer point cloud data set. Sewer defect classification	Four classes (normal, displacement, brick, and rubber ring)	DGCNN	OA = 47.9 mIoU = 46.1	17,027 points
			PointNet	OA = 18.4 mIoU = 18.5	
Yin et al. (2021)	Built a publicly available industrial indoor LiDAR data set. Proposed two neural modules to learn local structures	Six classes (I-beam, pipe, pump, rectangular beam, and tank)	PointNet	OA = 53.0 mIoU = 21.1	5 million points
			PointNet++	OA = 70.6 mIoU = 45.5	
			ResPointNet++	OA = 94.0 mIoU = 87.3	
Martin-Abadal et al. (2021)	Semantic segmentation of underwater pipe	Three classes (pipe, valve, and background)	PointNet	Mean $F1-\text{score} = 89.3$	262 points
Koo et al. (2021)	Classify infrastructure BIM elements	10 classes (column, 3 types of culverts, 5 types of walls, and sump)	PointNet	Mean $F1-\text{score} = 89.3$	1,496 elements
			PointNet	OA = 83 $F1-\text{score} = 87$	
Bahreini and Hammad (2021)	Detect concrete surface defect detection	Three classes (crack, spall, and no-defect)	DGCNN	OA = 98 F1 − score = 98	49 million points
Xia et al. (2022)	Bridge components segmentation	Two classes (slab and pier)	DGCNN	OA = 95.9 mIoU = 71.1	447 million points
			PointNet	OA = 84.4 mIoU = 45.9	

strategy to define an appropriate threshold considering RGB color values of point clouds and/or intensity values to detect defects. Two methods were used for damage detection: graph-based and surface normal-based methods. After applying those methods, clustering was used to group the defect points into individual defects. The aforementioned methods are sensitive to noise, uneven density, and complicated structures (Te et al. 2018). Valença et al. (2017) integrated image processing and laser scanning considering the RGB value for each point to measure the crack characteristics (e.g., orientation, width, and length).

On the other hand, several studies focused on detecting and quantifying spalls based on point clouds (Guldur Erkal and Hajjar 2017; Liu et al. 2010; Mizoguchi et al. 2013; Olsen et al. 2010; Teza et al. 2009). Olsen et al. (2010) proposed a slicing analysis to quantify the spalling volume of large-scale structural elements based on the cross-sectional slices of the point clouds. There are other approaches based on damage-sensitive features such as curvature (Teza et al. 2009), distance and gradient (Liu et al. 2010), and surface normal (Guldur Erkal and Hajjar 2017). An automated technique was proposed by Kim et al. (2015) to simultaneously localize and quantify spalling defects on concrete surfaces using TLS based on the angle and distance deviation of each point from the reference plane. The proposed technique was applicable for defects larger than 3 mm in depth and width. Guldur Erkal and Hajjar (2017) showed that data fusion between point clouds and vision data could lead to a promising solution to quantify the area and volume of the spalls.

Because they have an unordered nature, point clouds mostly are transformed into meshes (Kalfarisi et al. 2020) or voxels (Chen and Li 2016) used in machine learning (ML), particularly in supervised learning methods. Unlike images, ML-based methods for point cloud data sets have not made much progress for this purpose.

Some studies proposed a two-step approach, including applying the ML-based method to images for defect detection and then using point clouds for quantification of the detected defects (McLaughlin et al. 2020). Some methodologies were developed to reconstruct a bridge model using point clouds and then to detect and/or quantify the bridge surface defects using image processing (Chow et al. 2021; Kim et al. 2018). Turkan et al. (2018) proposed an adaptive wavelet neural network–based approach using point clouds to detect concrete surface defects. They concluded that this method was not practical for detecting hairline defects (Turkan et al. 2018).

The available information regarding the samples and results of the most relevant papers are summarized Table 2. Most of the current point cloud–based concrete surface defect detection methods focus on only one type of defects (Kim et al. 2015; Turkan et al. 2018). Focusing on one type of defect may not be practical, because inspectors try to detect different types and sizes of surface defects simultaneously.

Based on the literature review, not only the sizes and types of the defects but also the data acquisition conditions (e.g., the distance between the sensor and the damaged surface, sensor specifications, and resolution) vary from one paper to another. Therefore, the results of these papers are not comparable. Moreover, one of the major obstacles to comparing the results of those studies is the lack of unified evaluation metrics (e.g., accuracy, recall, and error). However, most ML methods are evaluated based on unified performance metrics such as accuracy, Intersection over Union (IoU), recall, and precision. The effect of different environmental conditions (e.g., lighting) cannot be ignored even if the same sensor settings have been used.

Objectives

To achieve an effective and efficient bridge visual inspection using a LiDAR scanner, this paper had the following objectives: (1) create a publicly available point cloud data set for concrete bridge surface defect semantic segmentation, and (2) develop a point cloud–based semantic segmentation DL method to detect different types of concrete surface defects.

Proposed Method

Aspects Considered in the Method

Because the PointNet++ semantic segmentation method originally was designed to detect indoor building elements, it cannot be applied in an off-the-shelf manner to the task of concrete surface defect detection. Therefore, four main aspects differentiate the proposed method, called Surface Normal Enhanced PointNet++ (SNEPointNet++), from the original PointNet++ method (Qi et al. 2017b):

Creating a Large High-Quality Point Cloud Data Set for Concrete Surface Defects

A sufficient point cloud data set is a key issue in the point cloud–based semantic segmentation of surface defects. Strict safety regulations and accessibility complications in scanning a bridge are the main reasons for the lack of point cloud data sets. Therefore, this study provides a high-quality point cloud data set to be used in surface defect detection by different groups of researchers. Furthermore, several data augmentation approaches, such as shifting, flipping, and rotating, can be applied to generate a larger data set based on the available data set. However, shifting is the least effective of these methods (Shijie et al. 2017). Moreover, using rotation may lead to altering the features of defects that are useful for learning and to creating defects that may not appear in actual structures. Therefore, in this work, the collected point cloud segments were augmented by flipping horizontally and vertically.

Redefining the Features of Points to Better Capture the Main Features of Defects

In the original PointNet++, the objects in each class are very similar. However, in defect detection, each defect has a unique shape, which is very different from those of other defects. These variations make the learning process difficult. Nevertheless, all cracks and spalls have two main features; they are deeper and darker than their adjacent areas. Therefore, if trained properly, the network can learn to use these features [i.e., point location (x_i, y_i, z_i) and color (R, G, B)] to distinguish between different types of defects and improve its performance.

In addition, to capture the curvature change of the surface, this study proposes using normal vectors as another input feature for training. The use of normal vectors for spall localization has been suggested in other works, such as by Kim et al. (2015), who developed a methodology based on the changes in the depth and the normal vectors of defects. However, those works did not use ML.

Let $F(x, y, z) = 0$ represent the surface, which is calculated based on a local surface fitting method. For a point $P_o = (x_o, y_o, z_o)$, the normal vector ($\vec{N}$) is computed using (Silverman 2002)

$$\vec{N} = \frac{\nabla F}{\nabla F} \tag{1}$$

where ∇F is the gradient of F; and ∇F is the vector length. Furthermore, in the original PointNet++ (Qi et al. 2017b), the normalized coordinate values of (X_i', Y_i', Z_i') also are used, which provide the network with the relative location of each point with respect to all other points in the segment. These parameters are computed based on (Qi et al. 2017a)

$$X_i' = \frac{x_i}{x_{\max}} \tag{2}$$

$$Y_i' = \frac{y_i}{y_{\max}} \tag{3}$$

$$Z_i' = \frac{z_i}{z_{\max}} \tag{4}$$

where $x_{\max}$, $y_{\max}$, and $z_{\max}$ = maximum values of x_i, y_i, and z_i in each segment, respectively. However, because the changes in depth indicate the possibility of defect existence, only Y_i' was used in this research. Removing X_i' and Z_i' helps the model to understand the contribution of depth in detecting surface defects. A reference plane matching the damaged surface is used to calculate the depth of the points of the defect (y_i) with respect to that surface (Fig. 1). The value of Y_i' increases with the depth of the defect. Moreover, ideally, the normal vectors of the deepest point of a defect ($\overrightarrow{N_1}$) and the nondefect points ($\overrightarrow{N_2}$) are perpendicular to the reference plane.

Fig. 2 shows a sample of blocks of points in a segment. Unlike the original PointNet++, in this research, 2D convolving for wrapping the data set is performed on the X, Z-surface (versus the X, Y-surface) using blocks with a given size of $A \times A$.

Addressing the Issue of Imbalanced Data for Priority Classes

The imbalanced class distribution was one of the challenges in this research, because not only are the number of defects in each class

Table 2. Summary of related point cloud–based concrete surface defect detection studies

	Type of defect					Features		Methodology			
Reference	Defect	Crack	Spall	Corrosion	Mass loss	Geometrical	Color and intensity	ML	Non-ML	Results	Size of defects
Teza et al. (2009)	Yes	—	—	—	—	Yes	—	—	PC	Accuracy (%): Circle (19–96) Rectangle (50–100) Square (50–100)	Defect sizes (cm): Circle ($r = 20$) Rectangle (10×20) Square (20×20)
Olsen et al. (2010)	—	—	Yes	—	—	Yes	Yes	—	PC	N/A	Spall size (cm^3): 8,700–24,500
Liu et al. (2011)	—	—	—	—	Yes	Yes	—	—	PC	N/A	Defect volumes (m^3): 2.71×10^{-3}–9.14×10^{-3} Defect areas (m^2): 4.27×10^{-2}–7.91×10^{-2}
Kim et al. (2015)	—	—	Yes	—	—	Yes	—	—	PC	Concrete panel (%): Accuracy: 68.3–91.3 Lab test (%): Recall: 49.8–92.3 Pre: 83.4–97.1	Spall size (mm^3): Min: $10 \times 10 \times 4$ Max: $100 \times 100 \times 7$
Truong-Hong et al. (2016)	—	Yes	—	Yes	—	Yes	Yes	—	PC	N/A	Dimensions (mm): W:4–12; L: 18–8
Guldur and Hajjar (2016)	Yes	—	—	—	—	Yes	Yes	PC	PC	Classification (%): Accuracy: 93.5 Pre: 94.5	N/A
Hou et al. (2017)	Yes	—	—	Yes	—	—	Yes	PC	—	Clustering error (%): Water staining: 10.1 Spall: 4.2	N/A
Guldur Erkal and Hajjar (2017)	—	Yes	Yes	—	—	Yes	Yes	PC	PC	Error (%): Crack: 0.15–8.2 Spall: 0.43–1.62	Dimensions (in.): Crack: L:1.9–12; W:1 Spall: L:11.7; W:3.7
Valença et al. (2017)	—	Yes	—	—	—	Yes	Yes	—	Img PC	N/A	Crack size (mm): L:30.1; W:1–4
Turkan et al. (2018)	—	Yes	—	—	—	Yes	—	PC	—	Error (mm): 1.4–2.8	Dimensions (mm): L:10–35; W:9–16
McLaughlin et al. (2020)	—	—	Yes	—	—	Yes	Yes	Img	PC	Error (%): 25.9	Area (cm^2): 394–3,277
Nasrollahi et al. (2019)	Yes	—	—	—	—	Yes	Yes	PC	—	Defect accuracy (%): 15–88 (Avg: 69)	Testing sample size (cm): Max depth: 1.5–8.1
Bahreini and Hammad (2021)	—	Yes	Yes	—	—	Yes	Yes	PC	—	Recall (%): Spall: 90; Crack: 55 Precision (%) Spall: 79; Crack: 55 IoU (%): Spall: 73; Crack: 45	Testing sample size (cm): Max depth: 1.5–8.1

Note: ML = machine learning; PC = point clouds; Img = image; L = length; and W = width.

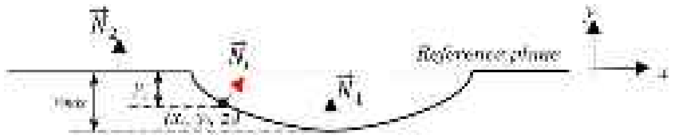


Fig. 1. Cross section of a sample defect with the normal vector and depth of a point.

(i.e., spalls and cracks) different, but their sizes also are very different. Moreover, the largest part of the data set belongs to the no-defect class, which has the least priority. Therefore, a weighted softmax cross-entropy loss function is applied to increase the contribution of the minority classes (spalls and cracks), which have priority. The weight (w_i) and the cost weight of class $i(W_i)$ are calculated using (Cui et al. 2019)

$$w_i = \frac{C_i}{\left(\sum_{i=1}^{K} C_i\right)} \tag{5}$$

$$W_i = \frac{1}{\log(1.05 + w_i)} \tag{6}$$

where C_i and K = number of points in class i and total number of classes, respectively.

Applying Sensitivity Analysis to Adjust the Hyperparameters to Better Capture the Features of Small Defects

Sensitivity analysis is applied to observe how the changes of each hyperparameter impacted the network's performance and to adjust them to better capture the features of small defects. The range of each hyperparameter is selected to cover the most promising values as follows:

1. Hyperparameters related to the network architecture
   a. Number of sublayers. Sublayers in PointNet++ are responsible for extracting the features from the sampled and grouped points using the mini-PointNet networks. The learning capacity of a neural network increases exponentially with its depth (i.e., number of sublayers) and polynomially with its width (i.e., number of nodes) (Montufar et al. 2014). Therefore, a sensitivity analysis of the number of sublayers is performed to obtain the optimal network topology. In addition, two sets of features are used in the input layer (with and without the normal vectors) to study the impact on performance.
   b. Sampling size. Having multiple sampling sizes is one of the advantages of PointNet++; it is expected that better results will be obtained using a variety of sampling sizes for different layers. Although all defects are relatively small, their sizes vary in a wide range. Therefore, the optimal sampling sizes should be able to accommodate large defects such as severe spalls as well as small cracks.
2. Hyperparameters related to the data set
   a. Size of the blocks and number of points in each block. The density of the data set depends on the number of points in each block and the block size (BL). If the density of a block is less or more than the predefined value, it will be up-sampled or down-sampled, respectively. In order to minimize the changes due to down-sampling and up-sampling, the number of points in each block and the block size must be adjusted considering the distribution of segments based on their densities. The original PointNet++ (Qi et al. 2017b) used 8,192 points in each 1.5×1.5-m block to make a uniform data set. Block sizes should be set based on the size of the segments and objects in each class (e.g., cracks and spalls). Therefore, it is expected that better results will be obtained using smaller blocks for defect semantic segmentation. Moreover, the number of points per block is selected based on the predefined block size so that the density of points fits within a reasonable density range. The density depends on the number of points and the area. Fig. 3 presents the process of defining the appropriate values for block size and number of points per block (NP). The initial values for the number of points per block (NP_0) is 8,192 points per block, and the initial values for block size (BL_0) should be smaller than the sizes of the smallest segment and the largest defect. In the first step, starting with NP_0 and BL_0, the density is calculated. If it is in the acceptable density range, which is defined based on the density of most of the segments to limit the changes in the data set due to down-sampling and up-sampling, NP_0 and BL_0 are considered to be an acceptable combination (COM). Then the other acceptable NPs for BL_0 are determined by increasing NP value by a certain value of $\alpha \times \text{NP}_0$ and the NP list is updated. If the density is not in the range, BL is decreased by $\beta \times \text{BL}_0$ and the acceptable NPs are selected out of the created NP list for the new BL. This process continues until the calculated density using the considered BL and NP_0 is out of the predefined density range (denoted OR).

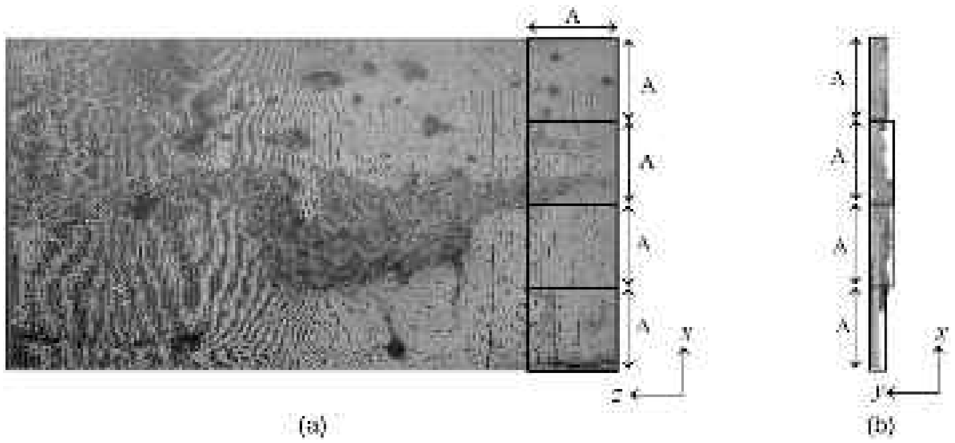


Fig. 2. Sample of blocks of points in a segment: (a) orthogonal view; and (b) cross section.

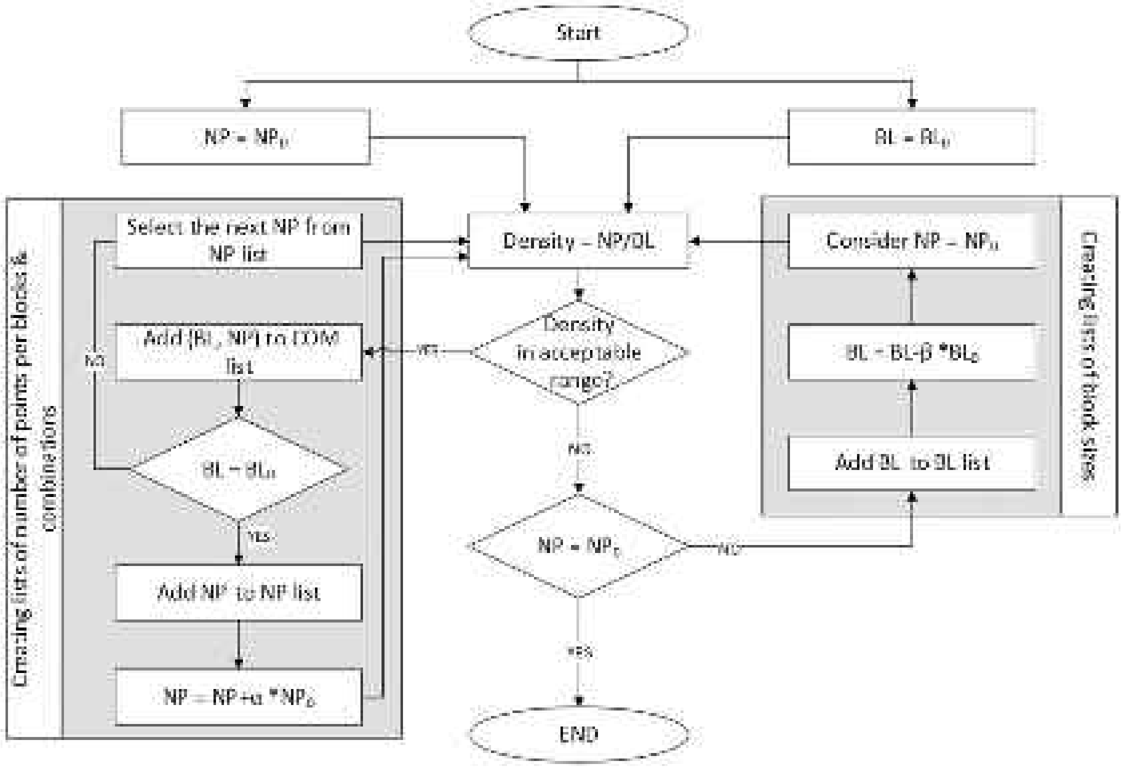


Fig. 3. Defining the values for sensitivity analysis for number of points per block and block size.

b. Stride size. Adding a stride in the data wrapping process may improve the results; however, it will increase the computation time. Therefore, sensitivity analysis helps to find the effect of the stride size on the results and select the most suitable value.

An efficient sensitivity analysis should be conducted in an organized multistep process. The sensitivity analysis is applied in four steps (Fig. 4). Step 1 aims to find the best architecture by varying the number of sublayers and the sampling sizes; the block size is set to its minimum value and the number of points per block is set to its maximum value because these values are expected to give the best results. For each number of layers, different combinations of sampling sizes are tried, and the performance metrics are calculated. Then, if it is the first case, a sublayer is added to investigate the performance of a deeper network. Otherwise, the results are compared with those of the previous models to determine if more layers are required and to select the best model.

After finding the best model in Step 1, the sensitivity analysis is applied to select the best block size out of the possible values, which are specified based on Fig. 3. In Step 2, the number of points per block and the stride size are fixed (i.e., maximum number of points and no stride). Based on the calculated performance metrics, the best value of the block size is selected. With the best model and block size determined, Step 3 focuses on obtaining the best value for the number of points per block in case of no stride. Finally, Step 4 investigates the effect of strides in different cases (i.e., 0%, 25%, and 50% strides).

Performance Metrics

Sensitivity analysis is applied, and the best combination of hyperparameters is selected based on five performance metrics, which are computed based on Eqs. (7)–(11) using the values of true positive (TP), false negative (FN), true negative (TN), and false positive (FP) for each class (e.g., crack)

$$\text{Precision} = \frac{TP}{(TP + FP)} \tag{7}$$

$$\text{Recall} = \frac{TP}{(TP + FN)} \tag{8}$$

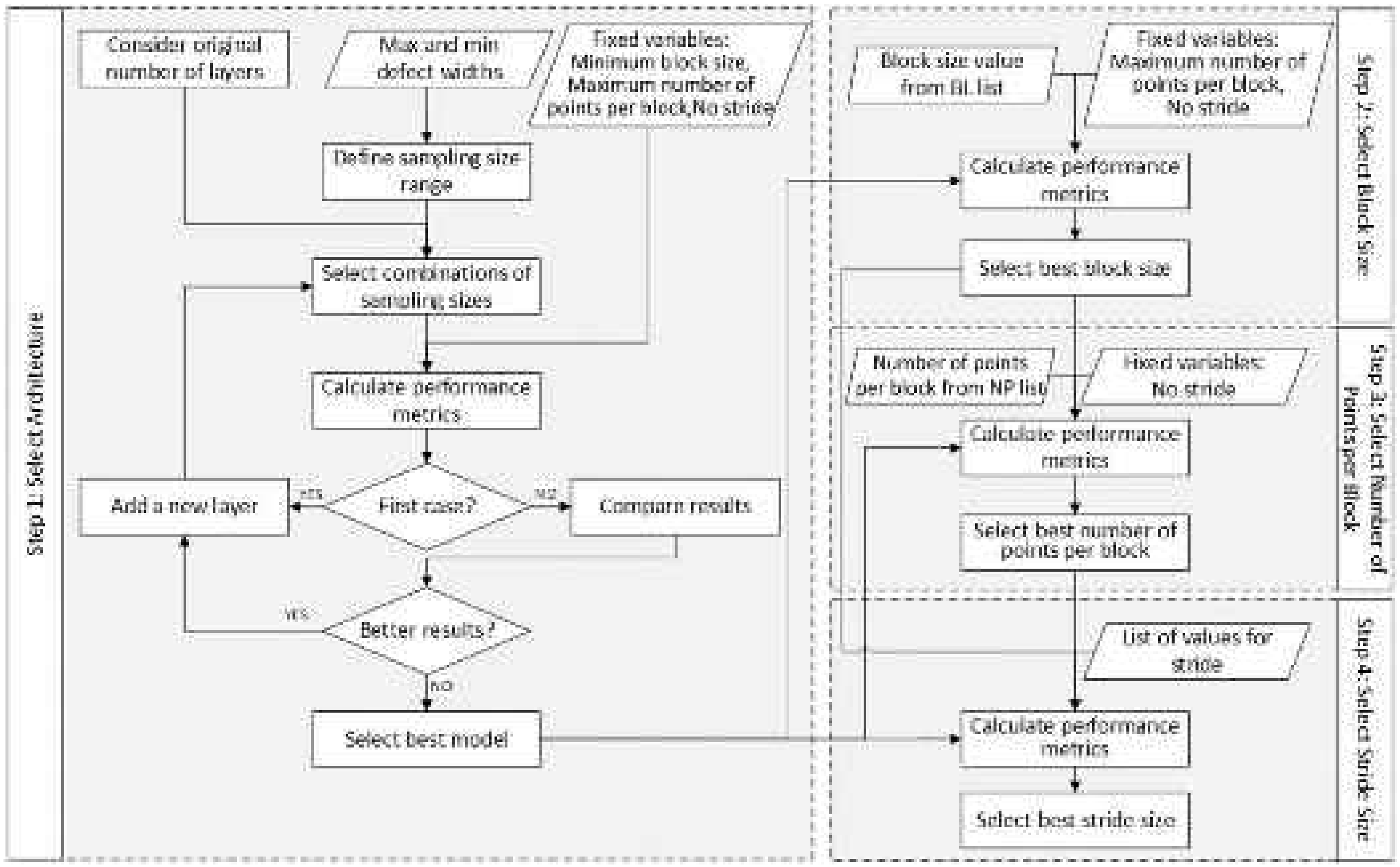
$$F1\text{-score} = \frac{2(\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})} \tag{9}$$

$$\text{IoU} = \frac{TP}{(FP + TP + FN)} \tag{10}$$

$$\text{Overall accuracy (OA)} = \frac{(\sum_{i=1}^{k} TP_i)}{(\text{number of points})} \tag{11}$$

where i and k = class number and total number of classes, respectively.

A higher TP leads to detecting more points correctly, which leads to higher precision and recall. If the focus is mainly on detecting most defects, recall has the priority to evaluate the network performance. For example, a network with higher cracks precision is able to predict more cracks with a smaller number of other defects (i.e., spalls and no defects) misclassified as cracks, whereas higher crack recall means missing fewer cracks and misclassifying fewer cracks. On the other hand, IoU is a commonly used metric to measure the similarity between the ground truth and the predicted region (Rahman and Wang 2016). Higher similarity requires minimizing FP and FN and maximizing TP. To ensure the efficiency of the proposed network, all these factors were considered in network performance evaluation. Most DL-based segmentation methods use simple loss functions (i.e., softmax) to optimize the OA (Rahman and Wang 2016).



Step 1: Select Architecture
Consider original number of layers
Max and min defect widths
Fixed variables: Minimum block size, Maximum number of points per block, No stride
Define sampling size range
Select combinations of sampling sizes
Calculate performance metrics
Add a new layer
YES
First case?
NO
Compare results
YES
Better results?
NO
Select best model
Block size value from BL list
Fixed variables: Maximum number of points per block, No stride
Calculate performance metrics
Select best block size
Step 2: Select Block Size
Number of points per block from NP list
Fixed variables: No stride
Calculate performance metrics
Select best number of points per block
Step 3: Select Number of Points per Block
List of values for stride
Calculate performance metrics
Select best stride size
Step 4: Select Stride Size

Fig. 4. Sensitivity analysis framework.

Framework of Adjustments Made to PointNet++

This research was conducted in two stages. The first stage, the adapted PointNet++, can be considered as a preliminary step; it focused on finding the best values for some hyperparameters such as stride sizes and the number of points per block without adding the normal vectors at each point as features. To this end, the data set was doubled by flipping it horizontally, and the network was fed by a seven-dimensional array including x, y, z, R, G, B, and Y'. This method is useful in the case of limited computation resources [i.e., limited random access memory (RAM)].

After evaluating the performance of the adapted PointNet++ and ensuring its feasibility, the aforementioned four aspects were applied in SNEPointNet++. After preprocessing the data set, SNEPointNet++ was trained based on the points represented by a 10-dimensional vector with values of x, y, z, R, G, B, N_x, N_y, N_z, and Y'. Moreover, sensitivity analysis covering all the hyperparameters in Fig. 4 was applied. Fig. 5 presents the overall framework of the proposed defect semantic segmentation; dashed boxes indicate the steps which are different between the two stages.

Implementation and Case Study

Fig. 6 shows the implementation and the case study process of defect semantic segmentation in detail. The initial steps, which mostly are related to removing irrelevant points and registration of the collected point clouds, are common between the adapted

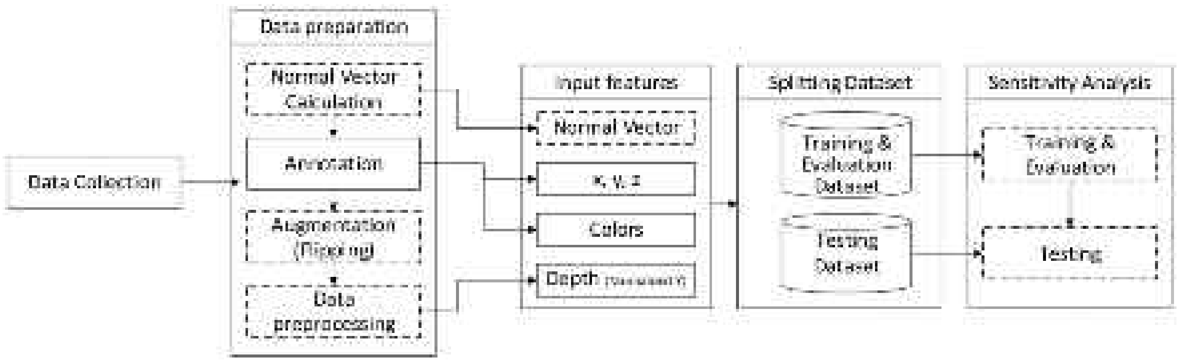


Fig. 5. Overall framework of the proposed defect semantic segmentation.

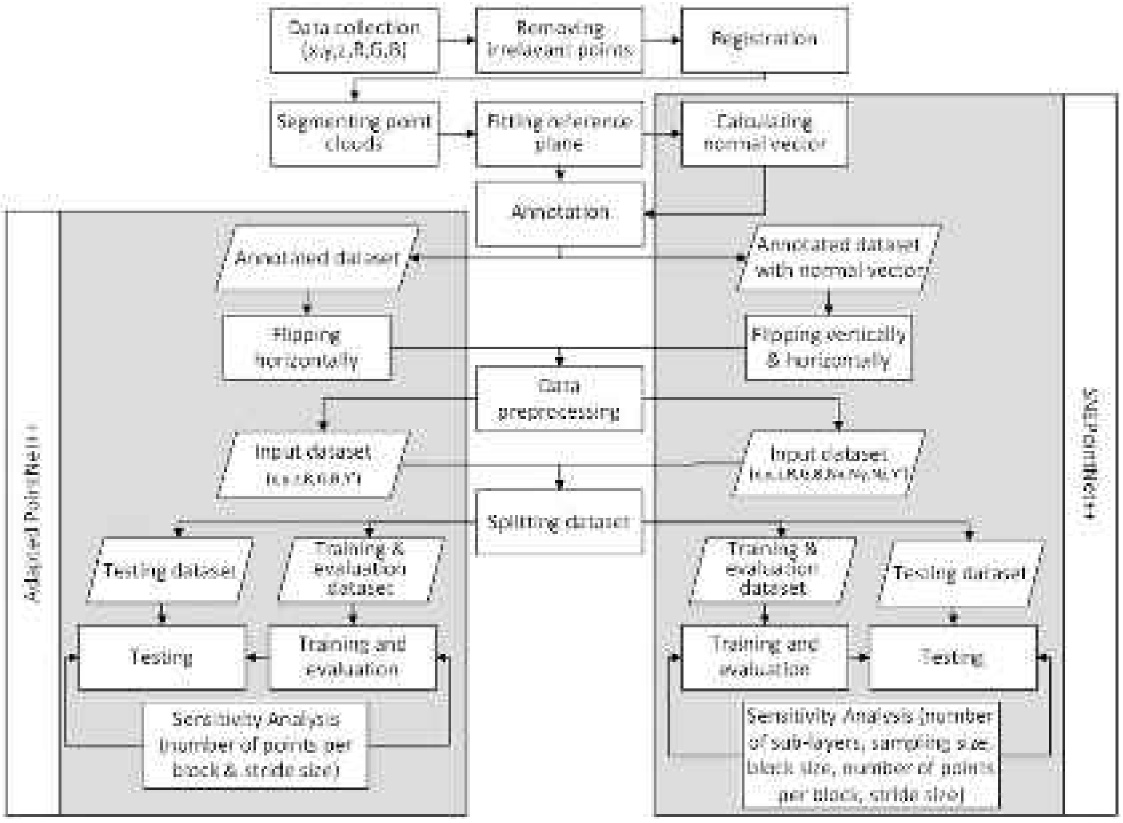

Data collection (x,y,z,R,G,B)
Removing irrelevant points
Registration
Segmenting point clouds
Fitting reference plane
Calculating normal vector
Annotation
Annotated dataset
Annotated dataset with normal vector
Flipping horizontally
Flipping vertically & horizontally
Data preprocessing
Input dataset (x,y,z,R,G,B,Y)
Input dataset (x,y,z,R,G,B,Nx,Ny,Nz,Y)
Splitting dataset
Testing dataset
Training & evaluation dataset
Training & evaluation dataset
Testing dataset
Testing
Training and evaluation
Training and evaluation
Testing
Sensitivity Analysis (number of points per block & stride size)
Sensitivity Analysis (number of sub-layers, sampling size, block size, number of points per block, stride size)
Adapted PointNet++
SNEPointNet++

Fig. 6. Implementation and case study steps of adapted PointNet++ and SNEPointNet++.

PointNet++ and SNEPointNet++. Although the process of some steps (i.e., annotation and data preprocessing) is the same for both networks, the inputs and outputs are different.

Data Collection

The inputs of the proposed method, which are the 3D point cloud data sets, were scanned from four RC bridges in Montreal using a FARO Focus3D X 130 scanner (FARO Technologies, Lake Mary, Florida). The 3D Faro laser scanner is equipped with a camera, which automatically captures images during scanning and detects the color of each point. The specifications of the scanner are presented in Table 3. Because scan planning (Aryan et al. 2021) was out of the scope of this research, planning the scanning stations and acquisition parameter settings for each bridge were not optimized. To guarantee that the acquired data sets fully covered the scanning targets including several parts of the bridge substructures (i.e., abutments), they were scanned from several stations. Fig. 7 shows samples of some scanning positions on the western side of Bridge 1. Increasing the number of stations may improve the data sets quality and increase the level of details due to more overlapping data sets. In addition, the scanner location impacts the incidence angle. A larger incidence angle gives a coarser data set. Therefore, scanning the same surface with different incidence angles may help to generalize the trained model for different intensities. Therefore, scanning Bridge 1 based on two different scan settings provided two data sets with different incidence angles and different qualities and colors. The acquisition parameters, such as FoVs, resolution, quality, and the number of scanned points, are presented in Table 4. Quality, which varies between 1× and 6×, is related to the length of time to capture points. For example, 1× and 4× denote 1 and 8 $\mu s/$scan point, respectively. Higher quality leads to longer scanning time and less errors (FARO Technologies Inc. 2017). The resolution is a factor to set the number of points acquired per rotation. It determines the density of the scan point, and can be between 1/32 and 1. For example, the Faro scanner can scan 710 million and 0.7 million points over a full area scan with its highest (1/1) and lowest (1/32) resolutions, respectively (FARO Technologies Inc. 2011). In other words, the resolution factor directly affects the distances between scanned points. For example, the minimum distances between two points using (1/1) and (1/2) resolutions are 0.3 and 0.62 mm, respectively, at 2-m scanning distance. Therefore, the minimum detectable widths of the defects which are located on a surface at a distance of 2 m in front of the scanner are 0.6 and 1.24 mm, respectively. In this case study, the minimum defect width (2 mm) was higher than these values. Although using the

Table 3. FARO Focus3D LiDAR scanner specifications

		Field of view (degrees)				
LiDAR	Points per second	Vertical	Horizontal	Angular resolution (degrees)	Accuracy (mm)	Measurement range (m)
FARO Focus 3D	976,000	305	360	0.009	±2	1.5–120 m

Source: Data from FARO Technologies Inc. (2011).

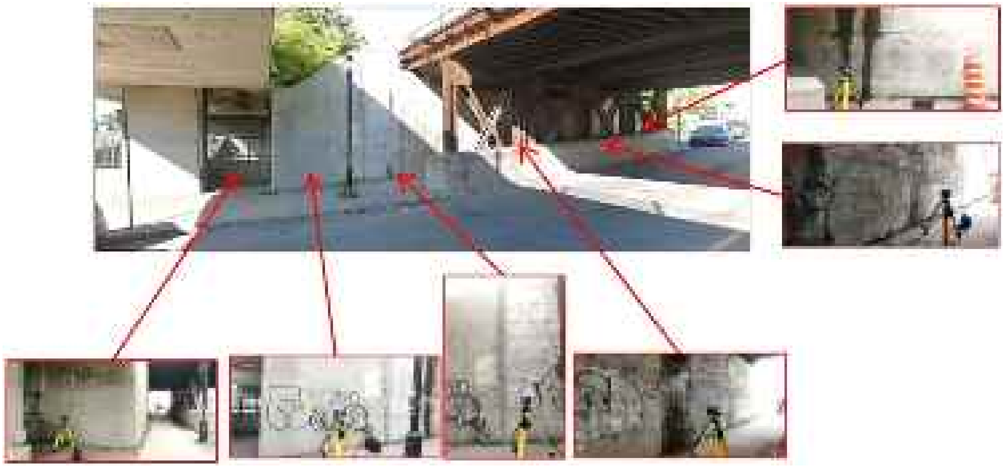


Fig. 7. Samples of scanning positions on the western side of Bridge 1.

Table 4. Scanning information

Parameters	Scan 1	Scan 2	Scan 3	Scan 4	Scan 5
Bridge number	1	1	2	3	4
Number of stations	8	4	6	4	2
Horizontal FoV (degrees)	23–259	23–259	0–360	0–360	0–360
Vertical FoV (degrees)	−43–71	−43–71	−63–90	−45–71	−60–90
Resolution	1/4	1/4	1/1	1/2	1/2
Quality	6×	6×	2×	4×	4×
Number of points (million)	25.5	25.5	710.7	134.5	177.7

highest resolution with maximum quality requires a longer scanning duration, it is required to minimize the errors. However, this is not always a feasible option due to battery limitations (i.e., battery capacity and performance in harsh weather), as well as traffic restrictions. To prevent scanning irrelevant objects, the horizontal and vertical FoVs can be decreased.

CloudCompare version 2.11 is 3D point cloud processing software. It recently has gained more popularity than some other software tools (e.g., Trimble software and ReCap) due to its free accessibility and higher speed in simple tasks such as registration and removing irrelevant data. However, due to an inconsistency between the output format of the Faro scanner (*.fls) and the input format acceptable by the CloudCompare software (*.ply), a format conversion was required, which was performed using ReCap software version 4.0.0.

Data Preparation

The first step in the data preparation was registering the scanned point clouds and removing the irrelevant points (i.e., buildings and trees). Then the scanned surfaces were divided into rectangular segments [Fig. 8(b)], which were large enough to contain different sizes of defects and small enough to avoid covering a large undamaged area. Using CloudCompare, the Z-axis of the canonical coordinate system was set in the vertical direction. The X-axis was along the concrete bridge surface, and the Y-axis was perpendicular to the concrete surface (outside direction) (Fig. 1). Normal vectors, which were considered only in SNEPointNet++, were calculated in CloudCompare software, and three attributes (N_x, N_y, and N_z) were added to each point (Bolourian 2022).

Because this study aimed to detect two types of concrete surface defects, cracks and spalls, the scanned point clouds were annotated manually and labeled into three classes: cracks, spalls, and no-defect. An example of an annotated segment is shown in Fig. 8(d). The statistical information of the combined data set is given in Table 5. The prepared data set contains 102 segments (about 27 million points) including 595 cracks (246,699 points) and 773 spalls (1,935,165 points). Fig. 9 shows the distribution of the data set based on segment density.

For the adapted PointNet++, the data set, which includes location and color features, was flipped horizontally. SNEPointNet++ used a triple-sized data set, including location, colors, and normal vector features, which was generated by flipping the original data set horizontally and vertically. The normalized value of $y(Y\prime)$ was calculated and added to each point. Then the annotated point clouds were wrapped inside the blocks based on the predefined values of block size, stride size, and the number of points in each block. To have a homogeneous data set, the number of points in all blocks was unified by down-sampling or up-sampling.

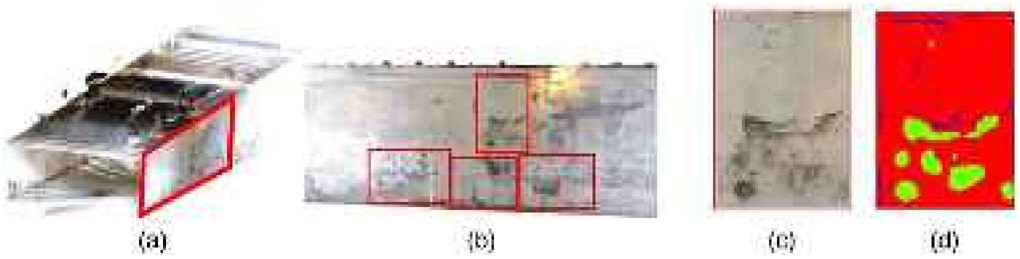


Fig. 8. Example of segmentation and annotation processes: (a) scanned bridge; (b) segmenting the bridge point cloud; (c) segment example; and (d) annotated segment.

Table 5. Statistics of annotated data set before augmentation

Data set	Number of segments	Number of points	Cracks		Spalls		No defect
			Number of cracks	Number of points	Number of spalls	Number of points	Number of points
Training and evaluation	81	21,313,285	475	182,430	588	1,252,551	19,878,304
Testing	21	5,628,620	120	64,269	185	682,614	4,881,737
Total	102	26,941,905	595	246,699	773	1,935,165	24,760,041

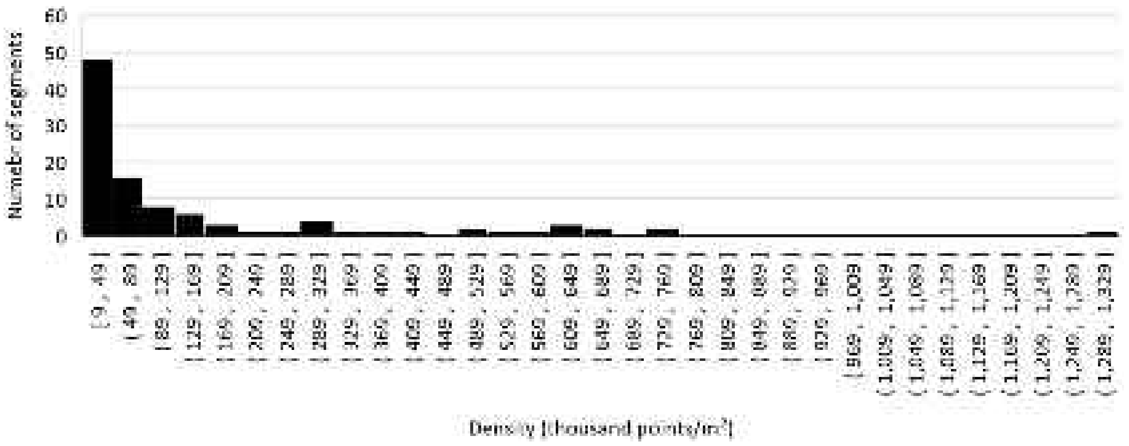


Fig. 9. Distribution of data set segments based on density.

Training and Testing

The method was implemented using TensorFlow-GPU 1.15.1, Cuda 11.0, and Python 3.6. The number of batches and initial learning rate were assumed to be 24 and 1×10^{-3}, respectively. The learning rate decayed 50% every eight epochs until the minimum value of 1×10^{-5} was reached. Based on the original PointNet++ (Qi et al. 2017b), rectified linear unit (ReLU) and softmax were used as activation functions, and Adaptive Moment Estimation (ADAM) was considered as the optimization method. Sensitivity analysis was applied to find the best combination of hyperparameters and adapt the network. The initial block size (BL_0) was 40 cm, which was smaller than the smallest dimension of segments (46 cm) and the largest defect size (60 cm) in this research. The density in each case should be between 9,000 and 329,000 pts/m^2, which is the density range for 85% of segments in the data set. Considering the acceptable density range and the computation resource limitations (i.e., RAM), three block sizes (i.e., 40×40 cm, 30×30 cm, and 20×20 cm) and three numbers of points per block (8,192, 10,240, and 12,288 points) were selected for sensitivity analysis.

Adapted PointNet++

Training and testing were performed on a cloud computing platform called Compute Canada (2021), using 2 GPUs, 24 GB RAM/GPU, and a 32-core CPU. Fig. 10 shows the adapted PointNet++ fed by seven-dimensional vectors. The hierarchical point set feature learning starts with sampling N_1 points of all n points of the data set in the first sublayer and considering each point as the centroid of a region. Then the points in a radius of 5 cm from the centroids are grouped and fed locally into a mini PointNet, which

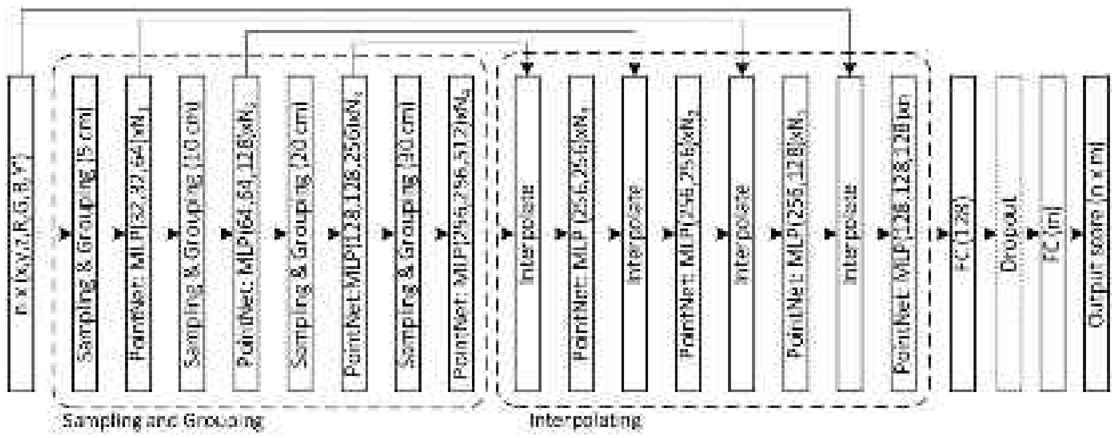

n × (x,y,z,R,G,B,Y)
Sampling & Grouping (5 cm)
PointNet: MLP(32,32,64)×N₁
Sampling & Grouping (10 cm)
PointNet: MLP(64,64,128)×N₂
Sampling & Grouping (20 cm)
PointNet: MLP(128,128,256)×N₃
Sampling & Grouping (30 cm)
PointNet: MLP(256,256,512)×N₄
Interpolate
PointNet: MLP(256,256)×N₃
Interpolate
PointNet: MLP(256,256)×N₂
Interpolate
PointNet: MLP(256,128)×N₁
Interpolate
PointNet: MLP(128,128,128)×n
FC(128)
Dropout
FC(m)
Output score (n × m)
Sampling and Grouping
Interpolating

Fig. 10. Architecture of adapted PointNet++.

comprises three multilayer perceptrons (MLPs) with 32, 32, and 64 nodes, followed by a max pooling operation to extract new features. The learning process continues considering 10, 20, and 30 cm radiuses for grouping the points around N_2, N_3, and N_4 number of centroid points, respectively, in three more sublayers. In the next phase, to interpolate the learning features, the inverse process is applied using four mini PointNet units, which are fed by two sets of features: (1) the features from the previous MLP, which are interpolated to fit to the current MLP; and (2) the features from the corresponding mini PointNet in the previous learning set (sampling and grouping). Finally, two fully connected (FC) layers are added to reach the score (m) for n points using the weighted softmax function.

In the preliminary sensitivity analysis, the hyperparameters were limited to the number of points per block and stride size. Nine cases (Cases A1–A9) were defined based on three numbers of points (i.e., 8,192, 10,240, and 12,288 points) and three stride sizes (i.e., 0%, 25%, and 50%) for the largest block size (40×40 cm) (Table 6). The results of training, evaluation, and testing are presented in Tables 6 and 7.

Changes in the number of points and stride sizes had effects on computation time (Fig. 11). Although the highest recalls for both cracks and spalls were in Case A9, training this model required the longest time (20 h 35 min) of all cases. Training Case A3 required 14 h less than training Case A9 and had only 2% and 0.2% decreases in crack and spall recalls, respectively. Considering 10,240 points/block with 50% stride (Case A8) resulted in the best performance in all terms except spall recall. Although increasing the number of points in Case A9 improved spall recall by 2.9% compared with Case A8 with no effect on crack recall, it decreased

Table 6. Training and evaluation results of adapted PointNet++ for various stride sizes and numbers of points per block

Case	Number of points per block	Stride size (cm)	Training		Evaluation	
			Mean loss	OA (%)	Mean loss	OA (%)
A1	8,192	40×40 (0%)	0.120	98.4	0.135	96.2
A2	10,240		0.087	98.7	0.102	96.8
A3	12,288		0.094	98.7	0.107	96.7
A4	8,192	40×30 (25%)	0.089	98.7	0.134	96.3
A5	10,240		0.079	98.8	0.114	96.7
A6	12,288		0.074	98.9	0.115	96.7
A7	8,192	40×20 (50%)	0.108	98.5	0.127	96.5
A8	10,240		0.071	**99.0**	0.100	**97.1**
A9	12,288		0.086	98.7	0.094	97.0

Note: Bold numbers are the highest values for OA in Training and evaluation.

Table 7. Testing results of adapted PointNet++ for various stride sizes and numbers of points per block (%)

Case	Cracks				Spalls				No defects			
	Precision	Recall	$F1$-score	IoU	Precision	Recall	$F1-$ score	IoU	Precision	Recall	$F1-$ score	IoU
A1	27.6	21.9	24.4	13.9	66.9	69.6	68.3	51.8	95.3	95	95.2	90.8
A2	38.4	45.2	41.6	26.2	67.6	77.5	72.2	56.5	96.5	94.4	95.5	91.3
A3	39.7	48.1	43.5	27.8	68.0	80.4	73.7	58.4	96.9	94.4	95.6	91.6
A4	29.5	26.5	27.9	16.2	75.8	75.7	75.7	60.9	96	96.2	96.1	92.5
A5	42.9	43.5	43.2	27.5	74.9	79.6	77.2	62.8	96.6	95.8	96.2	92.7
A6	41.7	49.3	45.2	29.2	76.6	75.6	76.1	61.4	96.2	96.1	96.1	92.6
A7	22.2	24.4	23.3	13.2	70.2	79.4	74.5	59.4	96.3	94.3	95.3	91
A8	**46.6**	**50.1**	**48.3**	**31.8**	**80.0**	77.7	**78.9**	**65.1**	96.3	**96.6**	**96.5**	**93.2**
A9	42.4	**50.1**	45.9	29.8	75.2	**80.6**	77.8	63.6	**96.9**	95.7	96.3	92.8

Note: Bold numbers are the highest value(s) of each column.

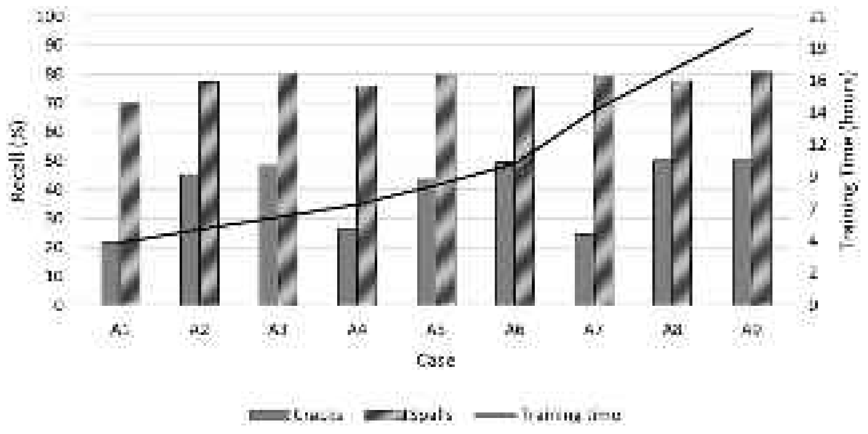


Fig. 11. Testing results and training time of Cases A1–A9.

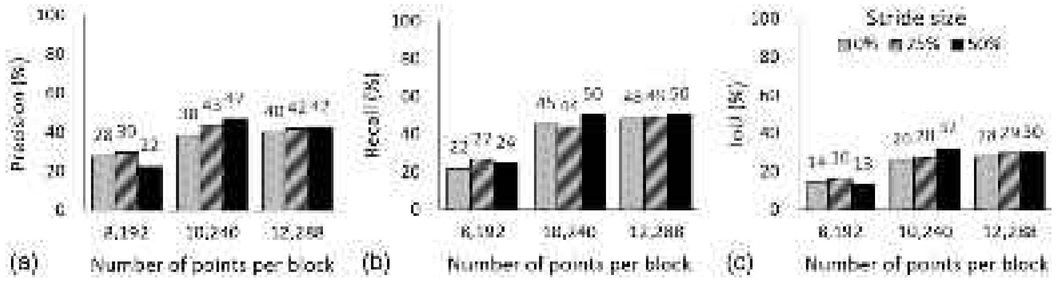


Fig. 12. Effect of stride and number of points on crack semantic segmentation using adapted PointNet++: (a) precision; (b) recall; and (c) IoU.

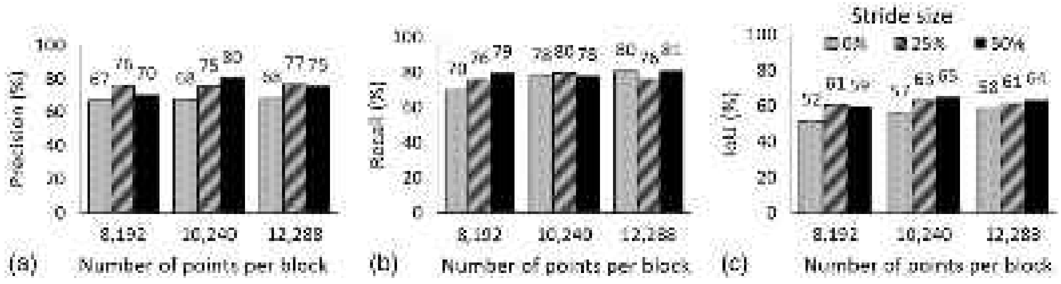


Fig. 13. Effect of stride and number of points on spall semantic segmentation using adapted PointNet++: (a) precision; (b) recall; and (c) IoU.

the time efficiency and performance of the network in terms of precision, $F1$-score, and IoU. Therefore, if there is a time limitation and the only focus is on not missing defects, Case A3 is a reliable choice for training a model. Otherwise, the overall performance of Case A8 is the best.

Fig. 12 presents the effect of stride and the number of points in each block on the adapted PointNet++ crack semantic segmentation. Adding the stride improved the network performance except in Case A5, which had a decrease of 1.7% in crack recall (Fig. 12). On the other hand, increasing the 25% stride to 50% for the blocks with 8,192 points decreased the efficiency of the model. In contrast, this change in strides caused improvement in the case of 10,240 points/block, whereas it had only minor impacts on training a model with 12,288 points. Therefore, the effect of strides on the overall performance of the model becomes less pronounced by increasing the number of points. This result indicates that by feeding the model with more points, the model reaches its maximum learning capacity, and the information provided by strides may be redundant or lead to overfitting.

Based on the outcomes of spall semantic segmentation using the adapted PointNet++ in Fig. 13, adding 25% stride improved the network performance in terms of IoU, recall, and precision, and only in the case of having 12,288 points did the spall recall decrease, by 4.8%. There was no clear pattern with respect to increasing the stride to 50%. The effect of changing the number of points was more prominent in the case of increasing 8,192 points to 10,240 points. For example, the greatest performance improvement was a 10% increase in precision with 50% stride and increasing the number of points from 8,192 points to 10,240 points.

SNEPointNet++

Training and testing of SNEPointNet++ were performed on a LAMBDA workstation with three NVIDIA RTX A6000 GPUs, 48 GB RAM/GPU, and an AMD Ryzer Threadripper 3960×48-core CPU. Most of the algorithms were developed in Python 3.8, and the environment was created in a Docker container. Table 8 presents the hyperparameter values in each step of the sensitivity analysis.

Table 8. Hyperparameter values in each round of sensitivity analysis

Hyperparameters	Step 1 (Series M)	Step 2 (Series B)	Step 3 (Series N)	Step 4 (Series S)
Sublayers and sampling size	Sublayers: 4, 5, 6 sampling sizes: 2.5–40 cm	Best of Series M	Best of Series M	Best of Series M
Block size (cm)	20 × 20	20 × 20 30 × 30 40 × 40	Best of Series B	40 × 40
Number of points per block	12,288	12,288	8,192 10,240 12,288	8,192 10,240 12,288
Stride size (%)	0	0	0	0 25 50

Effect of Number of Sublayers and Sampling Size

In Step 1, four, five, and six sublayers were considered to find the effective depth of SNEPointNet++. The widths of defects varied between 0.2 cm (i.e., hairline cracks) and 50 cm (i.e., severe spalls). Moreover, the smallest size of segments was 46 cm. Therefore, a relatively wide range of 2.5–40 cm was considered for the sampling size of each sublayer. The training and evaluation results and the training time for the five best networks are presented in Table 9, and their testing results are illustrated in Table 10. The results indicate that increasing the number of layers from four (M1 and M2) to five (M3 and M4) led to higher efficiency. Compared with the best five-layer network (M4), adding the sixth layer did not improve the learning process due to overfitting.

Considering five-layer networks, smaller sampling sizes (M4) resulted in about 2%–3% improvement over M3 in crack precision and IoU, respectively, and 0.8% and 0.6% reductions in recall and IoU of spalls, respectively. Considering the network performance in terms of both crack and spall semantic segmentation, M4 was selected as the most efficient architecture (Fig. 14).

Effect of Block Size and Number of Points per Block

In Step 2 and Step 3, two series of cases (B and N) were defined to adjust block size and number of points per block, respectively. To find the best combination of block size and number of points per block, first the block size and then the number of points per block were set. In Set B, considering three block sizes of 20×20 cm, 30×30 cm, and 40×40 cm with 12,288 points/block led to uniform densities of 307,200, 136,533, and 76,800 pts/m^2, respectively. Decreasing the block size from 40×40 cm to 30×30 cm resulted in better performance in terms of both cracks and spalls (Table 11). Although decreasing the block size from 30×30 cm (Case M4-B30) to 20×20 cm (Case M4-B20) improved the network performance for crack semantic segmentation, the spall recall and IoU decreased by 2.5% and 6.6%, respectively. It can be concluded that very small block sizes cannot be well-trained for larger defects (i.e., spalls). The most efficient block size was 20×20 cm due to its considerable effect on crack detection.

Table 12 presents the testing results of three cases defined based on different numbers of points (8,192, 10,240, and 12,288 points) per 20×20-cm block. More points per block resulted in better performance.

Effect of Stride Size

In Step 4, the stride could not be applied on the 20×20-cm blocks due to computation resource limitations (i.e., RAM size). Therefore, it was decided to consider the minimum possible block size

Table 9. Training and evaluation results of top five combinations of sublayers and sampling sizes

Case	Number of layers	Sampling size in each layer (cm)						Training		Evaluation		Training time (h:min)
		1	2	3	4	5	6	Mean loss	OA (%)	Mean loss	OA (%)	
M1	4	5	10	20	30	—	—	0.072422	98.1	0.073447	97.7	27:46
M2		2.5	10	20	30	—	—	0.080615	97.8	0.080318	97.5	28:03
M3	5	5	10	20	30	40	—	0.064733	98.3	0.064739	98.1	29:52
M4		2.5	5	10	20	30	—	0.058815	98.3	0.057952	98.1	30:35
M5	6	2.5	5	10	20	30	40	0.055237	98.3	0.056810	98.1	32:43

Table 10. Testing results of top five combinations of sublayers and sampling sizes (%)

Case	Cracks			Spalls			No defects		
	Precision	Recall	IoU	Precision	Recall	IoU	Precision	Recall	IoU
M1	67.4	90.6	63.9	86.3	90.1	78.8	99.2	98.5	97.7
M2	68.9	90.7	64.3	85.1	89.8	77.7	99.2	98.4	97.6
M3	71.2	91.3	66.6	90.4	91.2	**83.1**	99.4	98.5	98.0
M4	**73.3**	**93.0**	**69.2**	89.9	**92.0**	82.5	99.3	98.8	98.1
M5	70.9	93.6	67.6	**90.6**	90.5	82.8	99.2	98.8	98.1

Note: Bold numbers are the highest value(s) of each column.

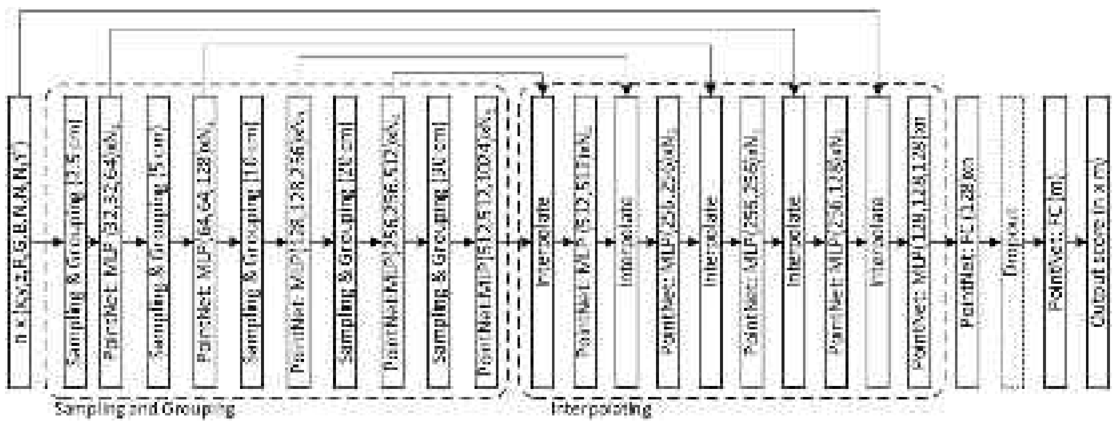

n × (x,y,z,R,G,B,N,N,N,Y)
Sampling & Grouping (2.5 cm)
PointNet: MLP (32,32,64)(xN)
Sampling & Grouping (5 cm)
PointNet: MLP (64,64,128)(xN)
Sampling & Grouping (10 cm)
PointNet: MLP (128,128,256)(xN)
Sampling & Grouping (20 cm)
PointNet: MLP (256,256,512)(xN)
Sampling & Grouping (30 cm)
PointNet: MLP (512,512,1024)(xN)
Interpolate
PointNet: MLP (512,512)(xN)
Interpolate
PointNet: MLP (256,256)(xN)
Interpolate
PointNet: MLP (256,256)(xN)
Interpolate
PointNet: MLP (256,128)(xN)
Interpolate
PointNet: MLP (128,128,128)(xn)
PointNet: FC (128)(xn)
Dropout
PointNet: FC (m)
Output score (n x m)
Sampling and Grouping
Interpolating

Fig. 14. Architecture of SNEPointNet++ with the best performance (Case M4).

Table 11. Testing results for different block sizes

			Cracks (%)			Spalls (%)			No defects (%)		
Case	Block size (cm)	Density (points/m^2)	Precision	Recall	IoU	Precision	Recall	IoU	Precision	Recall	IoU
M4-B20	20 × 20	307,200	**73.3**	**93.0**	**69.2**	89.9	92.0	82.5	99.3	**98.8**	**98.1**
M4-B30	30 × 30	136,533	61.7	81.5	54.2	**94.0**	**94.5**	**89.1**	**99.5**	98.4	97.9
M4-B40	40 × 40	76,800	44.4	62.3	34.9	84.5	86.4	74.6	99.0	98.6	97.6

Note: Bold numbers are the highest value(s) of each column.

Table 12. Testing results for different numbers of points per block

			Cracks (%)			Spalls (%)			No defects (%)		
Case	Number of points per block	Density (points/m^2)	Precision	Recall	IoU	Precision	Recall	IoU	Precision	Recall	IoU
M4-B20-N12	12,288	307,200	**73.3**	**93.0**	**69.2**	**89.9**	**92.0**	**82.5**	**99.3**	**98.8**	**98.1**
M4-B20-N10	10,240	256,000	62.2	85.8	57.9	83.0	87.7	76.2	99.3	98.5	97.5
M4-B20-N8	8,192	204,800	51.0	81.7	49.2	80.4	88.2	72.6	99.5	98.4	97.9

Note: Bold numbers are the testing results for the best case.

Table 13. Testing results for various stride sizes and numbers of points per block

			Cracks (%)			Spalls (%)			No defects (%)		
Case	Number of points per block	Stride size (cm)	Precision	Recall	IoU	Precision	Recall	IoU	Precision	Recall	IoU
M4-S1	8,192	40 × 40	17.0	74.8	16.1	85.7	90.3	78.5	**99.5**	98.6	98.1
M4-S2	10,240	(0%)	30.5	67.1	26.6	82.5	90.0	75.6	99.4	98.4	97.8
M4-S3	12,288		44.4	62.3	34.9	84.5	86.4	74.6	99.0	98.6	97.6
M4-S4	8,192	40 × 30	31.2	74.1	24.1	85.3	92.9	80.1	**99.5**	98.2	97.8
M4-S5	10,240	(25%)	42.9	76.4	37.9	82.8	92.4	77.6	99.4	98.2	97.7
M4-S6	12,288		44.0	77.1	38.9	86.1	92.3	80.3	99.4	98.6	98.0
M4-S7	8,192	40 × 20	36.0	72.5	31.7	86.4	**94.1**	82.0	**99.5**	98.5	98.0
M4-S8	10,240	(50%)	45.0	**79.7**	**40.4**	88.8	91.4	81.9	99.4	**98.9**	**98.3**
M4-S9	12,288		**45.8**	75.1	39.8	**89.1**	91.4	**82.3**	99.4	**98.9**	**98.3**

Note: Bold numbers are the highest value(s) of each column.

(40 × 40 cm) and investigate the effect of stride size (0%, 25%, and 50%) on the results for three different number of points per block (8,192, 10,240, and 12,288 points). The testing results of the SNEPointNet++ for nine cases are presented in Table 13.

Table 13 and Fig. 15 present the effect of increasing the stride size for different numbers of points in the case of crack semantic segmentation. By adding the strides (Cases M4-S4, M4-S5, and M4-S6), the performance mostly improved in terms of precision and IoU. These improvements were more dominant for the smaller number of points because there was more missing information, which can be provided using strides. Unlike IoU and precision, adding stride did not have any noticeable effect on recall for a small number of points (Case M4-S4), whereas it improved the performance for a larger number of points (Cases M4-S5 and M4-S6). The main reason for this trend may be the negative impact of increasing the number of points in the blocks with no stride (Cases M4-S1, M4-S2, and M4-S3). To evaluate the effect of increasing the stride size on model performance, Cases M4-S4, M4-S5, and M4-S6 were compared with Cases M4-S7, M4-S8, and M4-S9, respectively. The results demonstrated gradual improvements in crack IoU and precision, and minor fluctuations in recall.

Fig. 16 shows the testing results of spall semantic segmentation for Cases M4-S1–M4-S9. The greatest improvement due to adding strides was for Case S6 with 12,288 points, in which recall and IoU increased 5.9% and 5.7%, respectively. Although increasing the stride size from 25% to 50% resulted in a 6% increase in spall precision for Case M4-S8, it did not lead to a significant improvement in the performance, and in two cases the spall recall decreased. In general, compared with that for cracks, spall semantic segmentation performance was less sensitive to the stride size. Comparing the results of Case A8 in Table 7 with those of Case M4-S8 in Table 13 indicates that the modifications in the SNEPointNet++ (without decreasing the number of block sizes) led to 29.6% and 13.7% increases in crack and spall recalls, respectively, and in 8.6% and 16.8% performance improvement for crack and spall semantic segmentation, respectively, in terms of IoU.

Discussion

Training, Evaluations, and Testing Results

Table 14 summarizes the parameters and hyperparameters of the original PointNet++ and the two adjusted networks. The results of the training, evaluation, and testing of the best models of the adapted PointNet++ and SNEPointNet++ are presented in Tables 15 and 16. Using the adapted PointNet++ was useful as an initial step to ensure the feasibility of applying PointNet++ in this research domain. Compared with the adapted PointNet++, the performance of SNEPointNet++ in detecting cracks was 43% and 37% better in terms of recall and IoU, respectively. Moreover, in SNEPointNet++, the recall and IoU of spalls were 14% and 17% higher than in the adapted PointNet++, respectively. Furthermore, SNEPointNet++ improves the semantic segmentation performance of cracks more than spalls. The ground truth and the predicted results of three segments are visualized in Fig. 17. For Sample 1, most of the unpredicted crack points in the adapted PointNet++ were considered as no defects [Figs. 17(b and c)]. On the other hand, the normal vector can be a helpful factor in the learning process [Fig. 17(d)]. Other reasons for the improved results of SNEPointNet++ are the

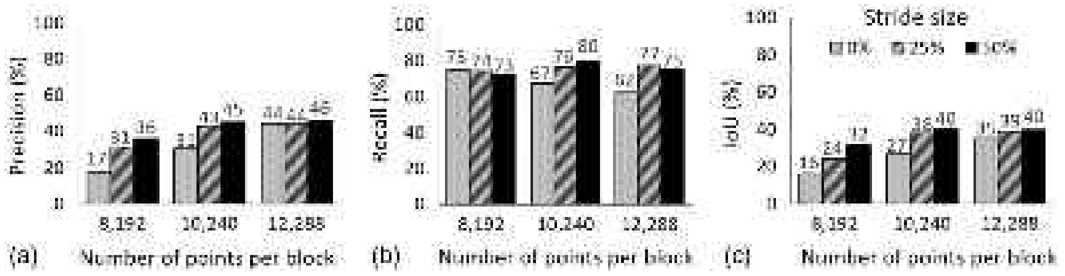


Fig. 15. Effect of stride and number of points on crack semantic segmentation using SNEPointNet++: (a) precision; (b) recall; and (c) IoU.

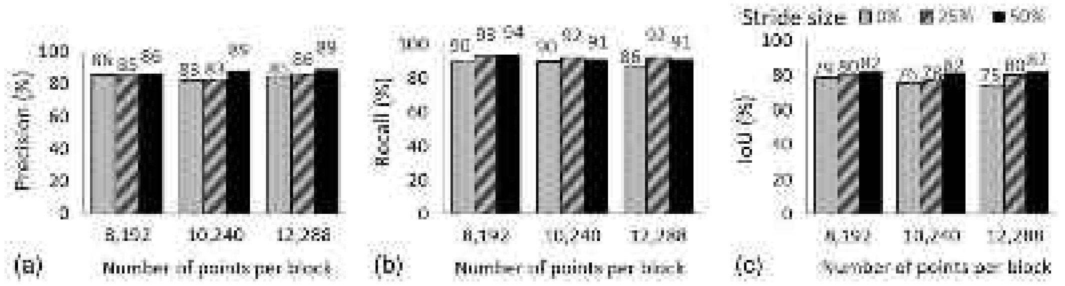


Fig. 16. Effect of stride and number of points on spall semantic segmentation using SNEPointNet++: (a) precision; (b) recall; and (c) IoU.

Table 14. Comparison of PointNet++ and adjusted networks (best configurations)

Parameter	PointNet++	Adapted PointNet++	SNEPointNet++
Classes	Building indoor objects (11 classes)	Cracks, spalls, no-defect	Cracks, spalls, no-defect
Data augmentation	Random rotation	Flipping horizontally	Flipping horizontally and vertically
Convolving direction	X, Y-surface	X, Z-surface	X, Z-surface
Size of blocks (m)	$1.5 \times 1.5 \times Z_{\max}$	$0.4 \times Y_{\max} \times 0.4$	$0.2 \times Y_{\max} \times 0.2$
Stride (%)	N/A	50	0
Input variables	$x, y, z, R, G, B, X', Y', Z'$	x, y, z, R, G, B, Y'	$x, y, z, R, G, B, N_x, N_y, N_z, Y'$
Number of points in each block	8,192	10,240	12,288
Number of sublayers	4	4	5
Sampling sizes (cm)	10, 20, 40, 80	5, 10, 20, 30	2.5, 5, 10, 20, 30
Number of epochs	200	50	50
Learning rate	1×10^{-3} (decays exponentially to minimum of 1×10^{-5})		

Table 15. Training and evaluation results

	Training		Evaluation	
Method	Mean loss	OA (%)	Mean loss	OA (%)
Adapted PointNet++	0.071	99.01	0.100	97.12
SNEPointNet++	0.059	98.3	0.058	98.1

deeper network and adding the smaller sample sizes (e.g., 2.5 cm) without removing other sample sizes. Hence, cracks have more chances to be detected.

Fig. 17(c) is a sample of network performance in differentiating between the defects and human-based color changes on the surface (i.e., inspectors' markings and graffiti). The network performed efficiently in detecting a line, which looks like a crack, as no defect. This is a good example of the advantage of point clouds over images.

Testing Results Based on Segment Depth

Testing results of the adapted PointNet++ (Case A8) and SNEPointNet++ (Case M4-B20-N12) are classified based on the maximum depth of each sample (segment) in Table 17. Despite the low recall for segments less than 5 cm deep, SNEPointNet++ modifications significantly improved the performance of semantic segmentation of deeper defects, especially cracks (Fig. 18).

Table 16. Summary of testing results (%)

	Cracks				Spalls				No defects			
Method	Precision	Recall	$F1$-score	IoU	Precision	Recall	$F1$ -score	IoU	Precision	Recall	$F1$-score	IoU
Adapted PointNet++	46.6	50.1	48.3	31.8	80.0	77.7	78.9	65.1	96.3	96.6	96.5	93.2
SNEPointNet++	**73.3**	**93.0**	**82.0**	**69.2**	**89.9**	**92.0**	**90.9**	**82.5**	**99.3**	**98.8**	**99.0**	**98.1**

Note: Bold numbers represent the highest value for each parameter.

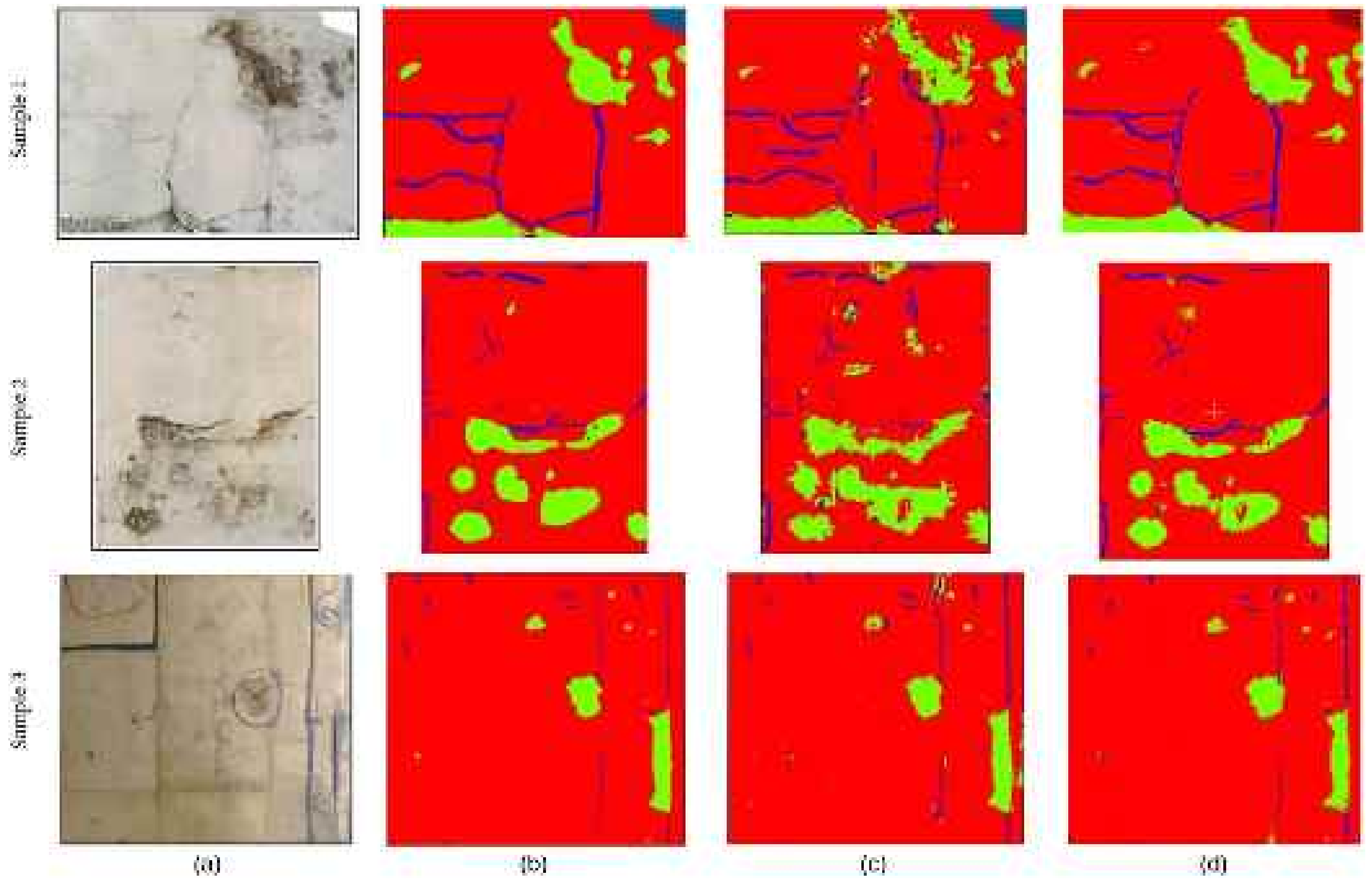


Fig. 17. Three samples of semantic segmentation: (a) original segment; (b) ground truth; (c) predicted (adapted PointNet++); and (d) predicted (SNEPointNet++).

Table 17. Classifying testing results based on segment depth

Depth, D (cm)	Defect type	Adapted PointNet++ Recall (%)	Adapted PointNet++ $F1$-score (%)	SNEPointNet++ Recall (%)	SNEPointNet++ $F1$-score (%)	Number of segments ratio (%)
$D \leq 3$	Cracks	11.8	15.7	42.6	36.2	9.5
	Spalls	69.9	71.33	79.9	79.8	
$3 < D \leq 5$	Cracks	12.0	15.3	65.3	53.9	14.3
	Spalls	67.6	69.0	82.2	76.0	
$5 < D \leq 7$	Cracks	54.0	51.3	92.3	89.3	42.9
	Spalls	66.9	75.3	85.7	89.7	
$7 < D$	Cracks	81.1	63.9	97.5	96.3	33.3
	Spalls	92.6	87.0	98.6	97.7	

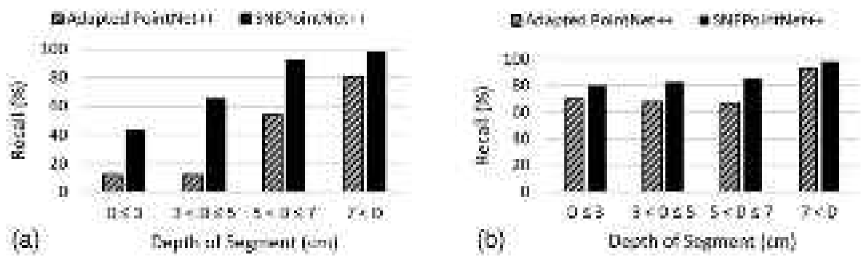


Fig. 18. Classifying testing results based on depth of segments: (a) cracks; and (b) spalls.

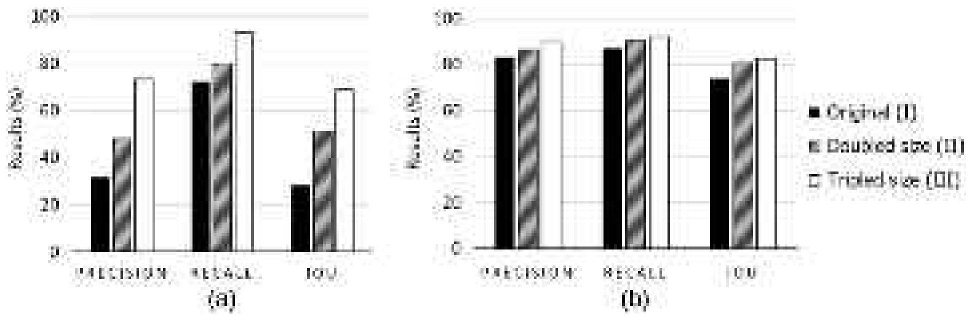


Fig. 19. Testing results based on three different data sets: (a) cracks; and (b) spalls.

The results show that the deepest segments had the highest recalls for both cracks (81% recall and 64% $F1$-score) and spalls (93% recall and 87% $F1$-score) using the adapted PointNet++. Using SNEPointNet++ improved the performance up to 98% and 99% recall for semantic segmentation of cracks and spalls, respectively. Moreover, the cracks of deeper segments were detected using SNEPointNet++ with 98% recall and 96% $F1$-score. These results validate the basic hypothesis of this research concerning the ability of point cloud–based DL methods to better detect defects by exploiting the depth information.

Effect of Data Augmentation

To investigate the effect of increasing the size of the data set by augmentation, three different data sets were used for training and testing: (1) the original data set, (2) the horizontally flipped (doubled size) data set, and (3) the horizontally and vertically flipped (tripled size) data set. Fig. 19 shows the performance metrics based on testing results. Increasing the size of the data set had a greater effect on crack semantic segmentation than on spall semantic segmentation. In crack semantic segmentation, using the horizontally flipped data set (Data set 1) led to 16% and 8% increases in precision and recall, respectively. Unlike spalls, augmenting the data set one more time (i.e., Data set 3 versus Data set 2) resulted in greater improvement in the case of cracks (25% precision and 13% recall). It can be concluded that in this stage, augmenting the data set can be more beneficial for crack feature learning than for spall feature learning.

Summary and Conclusions

This research makes two main contributions. Firstly, a point cloud–based DL method was developed for semantic segmentation of concrete bridge surface defects (e.g., cracks and spalls), called SNEPointNet++, considering the main features for defects (i.e., color, depth, and normal vector) and taking into account the issues related to the point cloud data set (i.e., smaller size of the data set, and imbalanced data set). Sensitivity analysis is applied to capture the best combination of hyperparameters and investigate their effects on network performance. Secondly, a point cloud data set is provided to detect two main concrete surface defects, spalls and cracks, and is publicly available for future research in concrete surface defect detection. The created high-quality point cloud data set includes 1,785 cracks and 2,319 spalls with a minimum width of 2 and 5 mm, respectively.

Based on the case study, it can be concluded that

1. In SNEPointNet++, using the main features related to surface defects (i.e., depth, color, and normal vectors) and taking into account the issues related to the point cloud data set (i.e., small size of the data set, and imbalanced data set) resulted in 93% (IoU = 69.2%) and 92% (IoU = 82.5%) recalls for semantic segmentation of cracks and spalls, respectively. Moreover, this network detected the spalls and cracks of the segments deeper than 7 cm, which had very severe defects, with 99% and 98% recall, respectively.
2. SNEPointNet++ with the best performance has five sublayers with 2.5, 5, 10, 20, and 30-cm sampling sizes.
3. Based on the sensitivity analysis, for the 20×20-cm blocks, more points per block results in higher efficiency. Moreover, considering 12,288 points/block, decreasing the block size improves the network performance for crack semantic segmentation due to increasing the density of each block, and consequently, providing more crack points. However, the optimum block size for spall semantic segmentation is 50% of the size of the largest defect, and using smaller blocks leads to a decrease in the efficiency of the network because the boundaries of spalls are missed in smaller blocks. Additionally, having strides improves the network performance in terms of IoU. However, high stride values are not beneficial in the case of a higher number of points per block (i.e., 12,288).
4. SNEPointNet++ is invariant to the resolution setting of the LiDAR scanner due to using different settings during data collection (i.e., 1/1, 1/2, and 1/4 resolutions). However, to detect cracks with widths as small as 2 mm, using the highest resolution is required.

Despite the aforementioned contributions, there are some limitations in this research that should be addressed in the future. These limitations are as follows:

1. One of the shortcomings of point cloud–based semantic segmentation in this research is the limited size of the data set. Therefore, future effort to increase the data set can focus on collecting more point clouds from different parts of the damaged bridges. Moreover, the point clouds can be generated using generative adversarial networks (GAN) (Li et al. 2018) and synthetic point clouds (Mohammadi et al. 2019).
2. The defect detection case study was limited to planar surfaces. Although this method was not evaluated on curved surfaces (e.g., columns), it is expected that it still will be applicable due to consideration of the normal vector. To evaluate the performance of SNEPointNet++, a data set including curved surfaces should be provided by scanning specific bridge elements such as circular columns or caps of piers. If the performance is not acceptable, considering the curvature as an input may improve the network performance.
3. One of the limitations in training SNEPointNet++ was the shortage of computation storage (i.e., RAM). In the future, having more resources will make it possible to expand the sensitivity analysis and improve the network performance. For example,

because increasing the number of points per block from 10,240 to 12,288 points significantly improved the network performance, the same trend is expected for larger numbers of points per block. Moreover, strides can be added to smaller blocks (e.g., 20 × 20 cm). In addition, other hyperparameters (i.e., depth and width of PointNet units, and number of points in each sample) can be adjusted in SNEPointNet++.

4. Inspectors require specific information about defects (i.e., size and severity). Therefore, future efforts should aim to classify the data set into more classes considering their levels of severity (i.e., medium or severe) or type of the cracks (i.e., shear cracks), although this approach will require more data in terms of variety and quantity; and extract the defects individually using supervised instance segmentation, clustering, or other non-ML methods, and then measure their dimensions.
5. SNEPointNet++ performance was validated based on an augmented data set including four features (i.e., 3D coordinates, colors, normal vector, and depth). However, the effects of each feature and each of the aspects, which were considered for model improvement, were not investigated separately. In the future, these aspects can be applied in separate steps to quantify their individual impact on the network performance. Moreover, future work will include comparing the performance of other semantic segmentation networks (i.e., DGCNN) when applied to concrete surface defects.

Data Availability Statement

Some or all data, models, or code generated or used during the study are available in a repository or online in accordance with funder data retention policies (https://github.com/neshatbln/SNEPointNet2).

Acknowledgments

The authors thank Ali Ghelmani for his help and contribution in implementation of SNEPointNet++.

References

Aryan, A., F. Bosché, and P. Tang. 2021. "Planning for terrestrial laser scanning in construction: A review." *Autom. Constr.* 125 (May): 103551. https://doi.org/10.1016/j.autcon.2021.103551.

Bahreini, F., and A. Hammad. 2021. "Point cloud semantic segmentation of concrete surface defects using dynamic graph CNN." In Vol. 38 of *Proc., 38th Int. Symp. on Automation and Robotics in Construction*, 379–386. Dubai, United Arab Emirates: IAARC Publication.

Bello, S. A., S. Yu, C. Wang, J. M. Adam, and J. Li. 2020. "Deep learning on 3D point clouds." *Remote Sens.* 12 (11): 1729. https://doi.org/10.3390/rs12111729.

Bolourian, N. 2022. "Surface normal enhanced PointNet2." Accessed May 12, 2022. https://github.com/neshatbln/SNEPointNet2.

Boulch, A., J. Guerry, B. Le Saux, and N. Audebert. 2018. "SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks." *Comput. Graph.* 71 (Apr): 189–198. https://doi.org/10.1016/j.cag.2017.11.010.

Chen, X., and J. Li. 2016. "A feasibility study on use of generic mobile laser scanning system for detecting asphalt pavement cracks." In *Proc., Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, ISPRS Arch*, 545–549. Göttingen, Germany: Copernicus Publications. https://doi.org/10.5194/isprsarchives-XLI-B1-545-2016.

Chow, J. K., K. Liu, P. S. Tan, Z. S. Su, J. Wu, Z. Li, and Y.-H. Wang. 2021. "Automated defect inspection of concrete structure." *Autom. Constr.* 132 (Dec): 103959. https://doi.org/10.1016/j.autcon.2021.103959.

Compute Canada. 2021. "Compute Canada." Accessed January 10, 2021. https://ccdb.computecanada.ca.

Cui, Y., M. Jia, T.-Y. Lin, Y. Song, and S. Belongie. 2019. "Class-balanced loss based on effective number of samples." In *Proc., IEEECVF Conf. on Computer Vision and Pattern Recognition. Recognition*, 9268–9277. New York: IEEE.

Dawood, T., Z. Zhu, and T. Zayed. 2017. "Machine vision-based model for spalling detection and quantification in subway networks." *Autom. Constr.* 81 (Sep): 149–160. https://doi.org/10.1016/j.autcon.2017.06.008.

Demir, N., and E. Baltsavias. 2012. "Automated modeling of 3D building roofs using image and LiDAR data." In *Proc., Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 35–40. Hannover, Germany: International Society for Photogrammetry and Remote Sensing.

Dorafshan, S., and M. Maguire. 2018. "Bridge inspection: Human performance, unmanned aerial systems and automation." *J. Civ. Struct. Health Monit.* 8 (3): 443–476. https://doi.org/10.1007/s13349-018-0285-4.

FARO Technologies. 2011. *User manual FARO laser scanner focus 3D*. Lake Mary, FL: FARO Technologies.

FARO Technologies. 2017. "Quality setting function on the Focus3D." Accessed May 24, 2021. https://knowledge.faro.com/Hardware/3D_Scanners/Focus/Quality_Setting_Function_on_the_Focus3D.

Gagnon, M., V. Gaudreault, and D. Overton. 2008. *Age of public infrastructure: A provincial perspective*. Ottawa: Statistics Canada.

Grilli, E., F. Menna, and F. Remondino. 2017. "A review of point clouds segmentation and classification algorithms." In *Proc., Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 339–345. Hannover, Germany: International Society for Photogrammetry and Remote Sensing.

Guldur, B., and J. F. Hajjar. 2016. "Automated classification of detected surface damage from point clouds with supervised learning." In Vol. 33 of *Proc., 34th Int. Symp. on Automation and Robotics in Construction (ISARC)*, 1–7. Taipei, Taiwan: IAARC Publications.

Guldur, B., Y. Yan, and J. F. Hajjar. 2015. "Condition assessment of bridges using terrestrial laser scanners." In *Proc., Structure Congress*, 355–366. Reston, VA: ASCE.

Guldur Erkal, B., and J. F. Hajjar. 2017. "Laser-based surface damage detection and quantification using predicted surface properties." *Autom. Constr.* 83 (Nov): 285–302. https://doi.org/10.1016/j.autcon.2017.08.004.

Guo, Y., H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. 2020. "Deep learning for 3D point clouds: A survey." *IEEE Trans. Pattern Anal. Mach. Intell.* 43 (12): 4338–4364. https://doi.org/10.1109/TPAMI.2020.3005434.

Haurum, J. B., M. M. Allahham, M. S. Lynge, K. S. Henriksen, I. A. Nikolov, and T. B. Moeslund. 2021. "Sewer defect classification using synthetic point clouds." In *Proc., 16th Int. Conf. on Computer Vision Theory and Applications*, 891–900. Setúbal, Portugal: SciTePress.

Hou, T.-C., J.-W. Liu, and Y.-W. Liu. 2017. "Algorithmic clustering of LiDAR point cloud data for textural damage identifications of structural elements." *Measurement* 108 (Oct): 77–90. https://doi.org/10.1016/j.measurement.2017.05.032.

Jiang, H., F. Yan, J. Cai, J. Zheng, and J. Xiao. 2020. "End-to-end 3D point cloud instance segmentation without detection." In *Proc., IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 12796–12805. New York: IEEE.

Kalfarisi, R., Z. Y. Wu, and K. Soh. 2020. "Crack detection and segmentation using deep learning with 3D reality mesh model for quantitative assessment and integrated visualization." *J. Comput. Civ. Eng.* 34 (3): 04020010. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000890.

Khaloo, A., D. Lattanzi, K. Cunningham, R. Dell'Andrea, and M. Riley. 2018. "Unmanned aerial vehicle inspection of the Placer River Trail Bridge through image-based 3D modelling." *Struct. Infrastruct. Eng.* 14 (1): 124–136. https://doi.org/10.1080/15732479.2017.1330891.

Kim, H., and C. Kim. 2020. "Deep-learning-based classification of point clouds for bridge inspection." *Remote Sens.* 12 (22): 3757. https://doi.org/10.3390/rs12223757.

Kim, H., J. Yoon, and S.-H. Sim. 2020. "Automated bridge component recognition from point clouds using deep learning." *Struct. Control. Health Monit.* 27 (9): e2591. https://doi.org/10.1002/stc.2591.

Kim, I.-H., H. Jeon, S.-C. Baek, W.-H. Hong, and H.-J. Jung. 2018. "Application of crack identification techniques for an aging concrete bridge inspection using an unmanned aerial vehicle." *Sensors* 18 (6): 1881. https://doi.org/10.3390/s18061881.

Kim, M. K., H. Sohn, and C.-C. Chang. 2015. "Localization and quantification of concrete spalling defects using terrestrial laser scanning." *J. Comput. Civ. Eng.* 29 (6): 1–12. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000415.

Kim, M.-K., H. Sohn, and C.-C. Chang. 2015. "Localization and quantification of concrete spalling defects using terrestrial laser scanning." *J. Comput. Civ. Eng.* 29 (6): 04014086. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000415.

Koch, C., K. Georgieva, V. Kasireddy, B. Akinci, and P. Fieguth. 2015. "A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure." *Adv. Eng. Inf.* 29 (2): 196–210. https://doi.org/10.1016/j.aei.2015.01.008.

Koo, B., R. Jung, Y. Yu, and I. Kim. 2021. "A geometric deep learning approach for checking element-to-entity mappings in infrastructure building information models." *J. Comput. Des. Eng.* 8 (1): 239–250. https://doi.org/10.1093/jcde/qwaa075.

Laefer, D. F., L. Truong-Hong, H. Carr, and M. Singh. 2014. "Crack detection limits in unit based masonry with terrestrial laser scanning." *NDT & E Int.* 62 (Mar): 66–76. https://doi.org/10.1016/j.ndteint.2013.11.001.

Lee, D., J. Kim, and D. Lee. 2019. "Robust concrete crack detection using deep learning-based semantic segmentation." *Int. J. Aeronaut. Space Sci.* 20 (1): 287–299. https://doi.org/10.1007/s42405-018-0120-5.

Li, C.-L., M. Zaheer, Y. Zhang, B. Poczos, and R. Salakhutdinov. 2018. "Point cloud GAN." Preprint, submitted October 13, 2018. https://doi.org/10.48550/arXiv.1810.05795.

Liu, W., S. Chen, D. Boyajian, and E. Hauser. 2010. "Application of 3D LiDAR scan of a bridge under static load testing." *Mater. Eval.* 68 (12): 1359–1367.

Liu, W., S. Chen, and E. Hauser. 2011. "LiDAR-based bridge structure defect detection." *Exp. Tech.* 35 (6): 27–34. https://doi.org/10.1111/j.1747-1567.2010.00644.x.

Maguire, M., S. Dorafshan, and R. Thomas. 2018. "SDNET2018: A concrete crack image dataset for machine learning applications." Accessed May 17, 2018. https://digitalcommons.usu.edu/all_datasets/48/.

Martin-Abadal, M., M. Piñar-Molina, A. Martorell-Torres, G. Oliver-Codina, and Y. Gonzalez-Cid. 2021. "Underwater pipe and valve 3D recognition using deep learning segmentation." *J. Mar. Sci. Eng.* 9 (1): 5. https://doi.org/10.3390/jmse9010005.

Maru, M. B., D. Lee, K. D. Tola, and S. Park. 2021. "Comparison of depth camera and terrestrial laser scanner in monitoring structural deflections." *Sensors* 21 (1): 201. https://doi.org/10.3390/s21010201.

Maturana, D., and S. Scherer. 2015. "Voxnet: A 3D convolutional neural network for real-time object recognition." In *Proc., IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 922–928. New York: IEEE.

McLaughlin, E., N. Charron, and S. Narasimhan. 2020. "Automated defect quantification in concrete bridges using robotics and deep learning." *J. Comput. Civ. Eng.* 34 (5): 04020029. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000915.

Mizoguchi, T., Y. Koda, I. Iwaki, H. Wakabayashi, Y. Kobayashi, K. Shirai, Y. Hara, and H. S. Lee. 2013. "Quantitative scaling evaluation of concrete structures based on terrestrial laser scanning." *Autom. Constr.* 35 (Nov): 263–274. https://doi.org/10.1016/j.autcon.2013.05.022.

Mohammadi, M. E., R. L. Wood, and C. E. Wittich. 2019. "Non-temporal point cloud analysis for surface damage in civil structures." *ISPRS Int. J. Geo-Inf.* 8 (12): 527. https://doi.org/10.3390/ijgi8120527.

Mohammed Abdelkader, E., O. Moselhi, M. Marzouk, and T. Zayed. 2021. "Entropy-based automated method for detection and assessment of spalling severities in reinforced concrete bridges." *J. Perform. Constr. Facil.* 35 (1): 04020132. https://doi.org/10.1061/(ASCE)CF.1943-5509.0001544.

Montufar, G. F., R. Pascanu, K. Cho, and Y. Bengio. 2014. "On the number of linear regions of deep neural networks." *Adv. Neural Inf. Process. Syst.* 27: 1–9. https://doi.org/10.48550/arXiv.1402.1869.

Nasrollahi, M., N. Bolourian, and A. Hammad. 2019. "Concrete surface defect detection using deep neural network based on LiDAR scanning." In *Proc, 7th Int. CSCE Conf.*, 1–10. Montreal: Canadian Society for Civil Engineering.

Olsen, M. J., F. Kuester, B. J. Chang, and T. C. Hutchinson. 2010. "Terrestrial laser scanning-based structural damage assessment." *J. Comput. Civ. Eng.* 24 (3): 264–272. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000028.

Pierdicca, R., M. Paolanti, F. Matrone, M. Martini, C. Morbidoni, E. S. Malinverni, E. Frontoni, and A. M. Lingua. 2020. "Point cloud semantic segmentation using a deep learning framework for cultural heritage." *Remote Sens.* 12 (6): 1005. https://doi.org/10.3390/rs12061005.

Qi, C. R., H. Su, K. Mo, and L. J. Guibas. 2017a. "PointNet: Deep learning on point sets for 3Dd classification and segmentation." In *Proc., IEEE Conf. on Computer Vision and Pattern Recognition*, 4–23. New York: IEEE.

Qi, C. R., L. Yi, H. Su, and L. J. Guibas. 2017b. "PointNet++: Deep hierarchical feature learning on point sets in a metric space." In *Advances in neural information processing systems*, 5105–5114. New York: Curran Associates.

Rahman, M. A., and Y. Wang. 2016. "Optimizing intersection-over-union in deep neural networks for image segmentation." In *Proc., Int. Symp. on Visual Computing*, 234–244. Cham, Switzerland: Springer.

Rashidi, M., M. Mohammadi, S. Sadeghlou Kivi, M. Abdolvand, L. Truong-Hong, and B. Samali. 2020. "A decade of modern bridge monitoring using terrestrial laser scanning: Review and future directions." *Remote Sens.* 12 (22): 3796. https://doi.org/10.3390/rs12223796.

Shijie, J., W. Ping, J. Peiyi, and H. Siping. 2017. "Research on data augmentation for image classification based on convolution neural networks." In *Proc., 2017 Chinese Automation Congress (CAC)*, 4165–4170. New York: IEEE.

Silverman, R. 2002. *Modern calculus and analytic geometry*. New York: Dover.

Te, G., W. Hu, A. Zheng, and Z. Guo. 2018. "Rgcnn: Regularized graph CNN for point cloud segmentation." In *Proc., 26th ACM Int. Conf. Multimedia*, 746–754. New York: Association for Computing Machinery.

Teza, G., A. Galgaro, and F. Moro. 2009. "Contactless recognition of concrete surface damage from laser scanning and curvature computation." *Non-Destr. Test. Eval. Int.* 42 (4): 240–249. https://doi.org/10.1016/j.ndteint.2008.10.009.

Truong-Hong, L., H. Falter, D. Lennon, and D. F. Laefer. 2016. "Framework for bridge inspection with laser scanning." In *Proc., 14th East Asia-Pacific Conf. on Structural Engineering and Construction (EASEC-14)*, 1–9. Ho Chi Minh City, Vietnam: EASEC.

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

Valença, J., I. Puente, E. Júlio, H. González-Jorge, and P. Arias-Sánchez. 2017. "Assessment of cracks on concrete bridges using image processing supported by laser scanning survey." *Constr. Build. Mater.* 146 (Aug): 668–678. https://doi.org/10.1016/j.conbuildmat.2017.04.096.

Wang, J., Y. Liu, X. Nie, and Y. L. Mo. 2022. "Deep convolutional neural networks for semantic segmentation of cracks." *Struct. Control Health Monit.* 29 (1): e2850. https://doi.org/10.1002/stc.2850.

Wang, Z., and F. Lu. 2016. "VoxSegNet: Volumetric CNNs for semantic part segmentation of 3D shapes." *IEEE Trans. Visual Comput. Graphics* 26 (9): 2919–2930. https://doi.org/10.1109/TVCG.2019.2896310.

Wu, Z., S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 2015. "3D ShapeNets: A deep representation for volumetric shapes." In *Proc., IEEE Conf. on Computer Vision and Pattern Recognition*, 1912–1920. New York: IEEE.

Xia, T., J. Yang, and L. Chen. 2022. "Automated semantic segmentation of bridge point cloud based on local descriptor and machine learning." *Autom. Constr.* 133 (Jan): 103992. https://doi.org/10.1016/j.autcon.2021.103992.

Yavartanoo, M., E. Y. Kim, and K. M. Lee. 2018. "SPNet: Deep 3D object classification and retrieval using stereographic projection." In *Proc., Asian Conf. on Computer Vision*, 691–706. Berlin: Springer.

Yin, C., B. Wang, V. J. L. Gan, M. Wang, and J. C. P. Cheng. 2021. "Automated semantic segmentation of industrial point clouds using ResPointNet++." *Autom. Constr.* 130 (Oct): 103874. https://doi.org/10.1016/j.autcon.2021.103874.

Zhang, W., Z. Zhang, D. Qi, and Y. Liu. 2014. "Automatic crack detection and classification method for subway tunnel safety monitoring." *Sensors* 14 (10): 19307–19328. https://doi.org/10.3390/s141019307.

Zhao, R., M. Pang, and J. Wang. 2018. "Classifying airborne LiDAR point clouds via deep features learned by a multi-scale convolutional neural network." *Int. J. Geogr. Inf. Sci.* 32 (5): 960–979. https://doi.org/10.1080/13658816.2018.1431840.