Development of Cracking Condition Assessment System for Concrete Bridge Decks Using Image Processing Techniques

ALDOT Project Number: 930-930

Submitted to:
Alabama Department of Transportation
1409 Coliseum Boulevard
Montgomery, Alabama 36110

Prepared by:

Shanglian Zhou
Wei Song, Ph.D.

Department of Civil, Construction, and Environmental Engineering,
College of Engineering,
The University of Alabama

December 31, 2021

Technical Report Documentation Page

1. Report No. (FHWA/CA/OR-)

2. Government Accession No.

3. Recipient's Catalog No.

4. Title and Subtitle
Development of Cracking Condition Assessment System for Concrete Bridge Decks Using Image Processing Techniques

5. Report Date:
January 31, 2022

6. Performing Organization Code

7. Author(s)
Shanglian Zhou, Wei Song

8. Performing Organization Report No.

9. Performing Organization Name and Address
Department of Civil, Construction and Environmental Engineering
The University of Alabama
260 H.M. Comer Hall
245 7th Avenue
Tuscaloosa, AL 35487

10. Work Unit No. (TRAIS)

11. Contract or Grant No.
ALDOT research Project No. 930-930 (GR 25663)

12. Sponsoring Agency Name and Address
Alabama Department of Transportation
1409 Coliseum Boulevard
Montgomery, Alabama 36110

13. Type of Report and Period Covered
Final Report
August 1, 2016 to December 31, 2020

14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

Modern society requires a sustainable, robust, and serviceable infrastructure system to promote social welfare and boost economy. To support such an infrastructure system, an efficient health monitoring framework is needed which can promptly detect the presence of defects and perform associated rehabilitation and maintenance. In civil infrastructure, one of the most common types of defects is cracking, which evolves rapidly under the impacts of heavy traffic, aging of materials, and drastic environmental changes. In recent decades, computer vision-based automated crack detection methodologies have been developed and extensively applied by professionals and researchers. Nevertheless, a few issues and challenges existing in this type of methodology are yet to be systematically investigated and properly addressed. In this report, a cracking condition assessment system is developed by leveraging advanced sensing and computer vision technologies, to address the issues and concerns in computer vision-based crack detection and provide accurate and efficient crack detection performance under real-world complexities. Experimental results and discussions show that the proposed cracking condition assessment system is capable to properly address the issues under investigation and leads to improved and more robust crack detection performance than current image-based methodologies.

17. Key Word(s)
Civil infrastructure, Computer vision, Crack identification, Deep learning, Heterogeneous image fusion

18. Distribution Statement

19. Security Classif. (of this report)
Unclassified

20. Security Classif. (of this page)
Unclassified

21. No. of Pages

22. Price

TABLE OF CONTENTS

List of Tables

List of Figures

Executive Summary

Modern society requires a sustainable, robust, and serviceable infrastructure system to promote social welfare and boost economy. To support such an infrastructure system, an efficient health monitoring framework is needed which can promptly detect the presence of defects and perform associated rehabilitation and maintenance.

In civil infrastructure, one of the most common types of defects is cracking, which evolves rapidly under the impacts of heavy traffic, aging of materials, and drastic environmental changes. In recent decades, image-based automated crack detection methodologies have been developed and extensively applied by professionals and researchers. Nevertheless, a few issues and challenges existing in this type of methodology are yet to be systematically investigated and properly addressed.

In this technical report, an image-based condition assessment framework for roadway crack detection is developed. It consists of four topics: *i*) proposing a filter-based methodology that can address image disturbances to promote a robust image-based roadway crack detection; *ii*) performing a systematic study to investigate the impact from hyperparameter selection on the performance of deep convolutional neural network (DCNN) on roadway crack classification; *iii*) achieving pixel-level crack detection resolution on image data of real-world complexities through DCNN-based roadway crack segmentation; and *iv*) investigating the impact from heterogeneous image data on DCNN-based roadway crack detection and proposing heterogeneous image fusion strategies to address data uncertainties.

Overall, experimental results and discussions show that the proposed crack detection framework is capable to properly address the issues under investigation and leads to improved and more robust crack detection performance than current image-based methodologies.

CHAPTER 1: INTRODUCTION

1.1 Motivation

Recent years have witnessed a steady trend of growth in the transportation infrastructure investment. For example, according to the U.S. Census Bureau [1], the total construction spending on highway and street during March 2020 was estimated at a seasonally adjusted annual rate (SAAR) of 108.6 billion dollars, 4.5% higher than in February 2020 and 5.3% higher than in March 2019. Over a wider time span, the SAAR of the construction spending on highway and street has steadily increased from 78.2 billion dollars in March 2010 [2] to 108.6 billion dollars in March 2020 [1], at an annualized growth rate of 3.3%. Alongside the vigorous development of the transportation infrastructure, there has been a rising demand for a more efficient investment on transportation infrastructure by facilitating performance-based decision-making and appropriately managing infrastructure assets for better stewardship of the transportation system.

The Moving Ahead for Progress in the 21st Century Act (MAP-21) [3] sets the course for transportation investment in highways to address many challenges facing the U.S. transportation system such as improving safety and maintaining infrastructure condition. One of the core concepts and requirements under MAP-21 is to establish performance-based planning and programming to promote an efficient transportation system and improve transportation decision-making. For the Federal Highway Administration (FHWA), State Department of Transportation (DOT), and local governments, long-term efforts need to be devoted to implementing performance management activities and methodologies, such that the requirements by MAP-21 are fulfilled.

Right-of-way (ROW) imagery has become one of the data sources submitted to both the Transportation Asset Management Plans (TAMP) [4] mandated by MAP-21 and the Highway Performance Monitoring System (HPMS) [5]. By using image data taken from a ROW imaging system (e.g., a survey vehicle), image-based methodologies with advanced computer vision techniques for roadway defects detection can offer an efficient performance-based framework to facilitate the condition assessment and decision-making process for transportation infrastructure. Thus, it has been a rising interest for professionals and researchers to develop more efficient and accurate image-based methodologies on roadway defects detection for the purpose of promoting social welfare and stimulating economy through improving the transportation system.

1.1.1 *Literature review on image-based crack detection*

Cracking distress is one of the most common types of defects in civil infrastructure [6], which is caused by various factors such as vehicle loading, aging of materials, and long-term environmental effects, etc.[7] For example, the freeze-thaw effect [8,9] can cause concrete cracking due to moisture freezing and expansion; the alkali-silica reaction (ASR) can lead to aggregates expansion and cracking [10]. As such cracks gradually evolve, propagate, and coalesce into major structural cracks, the integrity and serviceability of the infrastructure (e.g., bridges [11,12], roadways [13,14], pipelines [15], and tunnels [16]) is largely deteriorated, which is one of the pressing concerns facing today's civil engineering community. Taking bridge structures as an example, cracks can not only impair the bridge aesthetics but also cause structural damage by allowing corrosive chemical agents (e.g., water and de-icing salts) to penetrate through the bridge decks, causing deteriorated serviceability and even structural failures. Thus, the development of accurate and efficient crack detection methodologies using imagery data is of great importance, as it facilitates a prompt transportation decision-making and rehabilitation by detecting and providing cracking information for health monitoring and condition assessment of the infrastructure.

In recent decades, image-based crack detection methodologies have been extensively developed and applied [6]. Compared against some traditional manual approaches such as visual inspection [17], which are usually subjective and labor-intensive [18], image-based methodologies offer an automated yet more consistent and objective alternative that can reduce labor cost and improve crack detection efficiency.

From the perspective of feature representation and classification, current image-based crack detection methods can be categorized into non-learning-based and learning-based methods. Most of early image-based methods belong to the non-learning-based category. This type of methodology usually employs handcrafted image processing techniques such as filtering, thresholding, and morphological operation, for crack enhancement and cracking feature extraction. Although certain levels of success were reported in their applications, the non-learning-based methods still suffer from some issues. For example, the handcrafted image processing procedures or techniques used in non-learning-based methodologies are usually involved with prior user input or subjective parameter selection. Thus, their applicability under real-world scenarios is usually limited, due to the subjectivity from human intervention and the lack of ability to self-adapt to variations of the ambient environment.

Learning-based methods can potentially alleviate the above subjectivities by directly learning from data, making predictions via self-learned pattern recognition and extraction. In the recent decade, deep convolutional neural network (DCNN), as a type of deep learning-based method, has rapidly evolved into the most advanced and popular crack detection methodology due to its high versatility and wide applicability. Nevertheless, for current DCNN-based crack detection methodologies, several issues and challenges related with the DCNN layout and image data also exist.

1.2 Challenges

The major challenges for today's image-based crack detection methodologies are briefly summarized as follows, a majority of which remain to be systematically investigated and properly addressed to promote the robustness of image-based methodologies against real-world data:

i) Challenge 1 (disturbances in intensity images): For crack detection methodologies using intensity image data, the general assumption is that cracks are darker (i.e., lower intensity value) than non-crack regions [19]. However, during image data collection, influences from ambient environment such as shadows, varying lighting condition [19-23] often exist in the acquired intensity data, which can result in non-uniform background illumination and low intensity contrast, thus deteriorating the crack detection performance. Besides, image disturbances such as stains, oil spills, and tire marks [19,24-26] which possess crack-like features and characteristics can add difficulties to intensity-based crack detection methodologies. Figure 1-1 illustrates some typical types of disturbances existing in intensity image data;

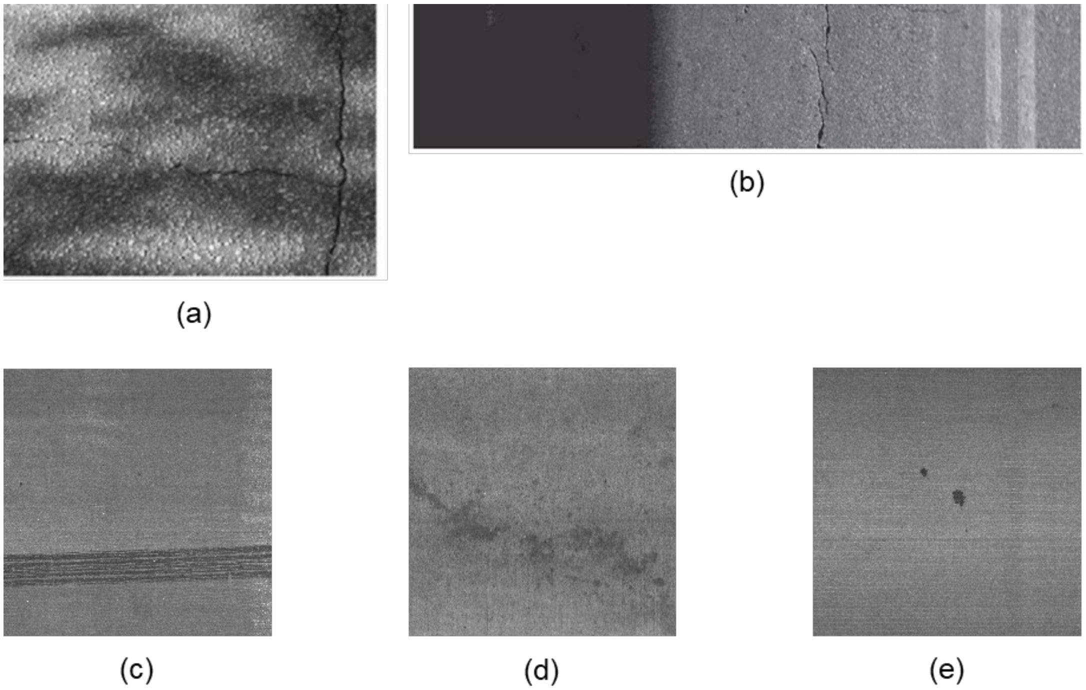


Figure 1-1. Disturbances in intensity images: (a) shadows [27]; (b) uneven illumination [28]; (c) tire mark; (d) oil spill; and (e) stains.

ii) Challenge 2 (disturbances in range images): Crack detection methodologies using range (i.e., elevation) image data usually rely on the elevation difference between cracks and non-crack regions to interpret the presence of cracks [19]. Despite its advantages over intensity image data such as being insensitive to varying lighting condition, range image data also suffers from disturbances such as surface variations, grooved patterns, shoulder drop-offs, pavement joints, and pavement edges [29-31]. These image disturbances are undesirable, because they usually lead to false positive detections due to similar elevation changes as cracks. Moreover, the performance of range-based methodologies may deteriorate on cracks with shallow depths [29,31]. Examples on the disturbances in range images are illustrated in Figure 1-2;

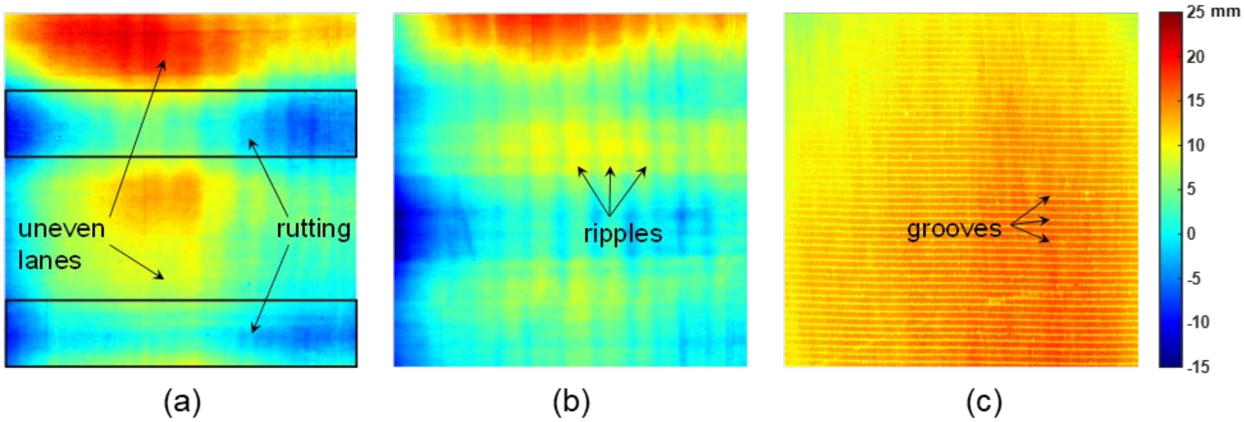


Figure 1-2. Disturbances in range images: (a) rutting and uneven lanes; (b) ripples due to vehicle vibration; and (c) grooved patterns [29].

iii) Challenge 3 (subjectivities due to parameter selection in image pre-processing techniques): The image disturbances existing in intensity or range image data, as introduced above, need to be properly addressed prior to crack detection for crack enhancement and noise suppression purpose. For non-learning-based methodologies, it is common practice to apply image pre-processing techniques such as filtering [13,26,32,33], morphological operation [34,35], and surface fitting [36] to eliminate these image disturbances from the image data. However, such image pre-processing techniques are usually involved with a parameter selection process, which is often expertise-intensive, user-dependent, and case-dependent. The subjectivities and uncertainties associated with the image pre-processing techniques usually exist, leading to performance deterioration of the crack detection methodologies against image data of real-world complexities;

iv) Challenge 4 (detection of crack objects with enclosed boundaries): The majority of the non-learning-based crack detection methodologies such as edge-based crack detection usually do not consider crack connectivity [37], thus have difficulties estimating the cracking properties such as area and width which may serve as important indicators for health condition assessment and decision-making;

v) Challenge 5 (hyperparameter configuration for DCNN-based crack detection): For crack detection applications based on DCNN, a critical factor on the network performance is to determine the optimal joint configuration of the hyperparameters such as network width, depth, and learning rate [31]. Some hyperparameters such as network depth, width, and number of filters determine the DCNN layout and, accordingly, the model capacity and complexity. Furthermore, other hyperparameters such as learning rate and momentum factor governs the efficiency of the DCNN on learning from the input data [38]. Determination of these hyperparameter values is usually an experimental process, and no exact guidelines to the design process of an ideal DCNN architecture and its associated training scheme for crack detection tasks are available [38,39];

vi) Challenge 6 (pixel-level crack classification on real-world image data through DCNN): In recent years, DCNN-based methodologies have gradually evolved from patch-level crack classification to crack object detection and then to pixel-level classification (i.e., segmentation). Semantic segmentation through DCNN can realize pixel-level resolution on crack detection. Nevertheless, a few issues remain to be properly addressed for this type of methodology. For example, due to the uncertainty in real-world image data, DCNN-based crack segmentation studies such as [30,40-42] often employ image pre-processing techniques to address image-related issues, despite the high adaptability of DCNNs. However, it is preferable to exploit raw image data rather than pre-processed data to minimize human intervention. Moreover, current methodologies [30] have difficulties achieving robust crack segmentation performance on raw image data contaminated by disturbances such as grooved patterns in range images;

vii) Challenge 7 (DCNN-based crack detection on real-world image data through data fusion): For DCNN-based crack detection applications, either intensity or range image data is used for analysis. As explained above, the image-related issues existing each type of image data need to be properly addressed through image pre-processing techniques even for DCNN-based methodologies [30,40-42]. The uncertainties and subjectivities associated with the image pre-processing, however, cannot be completely avoided. From the perspective of image data, it may be feasible to exploit data fusion strategies [43-45] to combine the information in different types of image data to address data uncertainties and obtain integrated and comprehensive information, as a better alternative to applying image pre-processing techniques. Nevertheless, in current literature on DCNN-based crack detection, studies and discussions on developing effective data fusion strategies are lacking; besides, the influences from different types of image data on DCNN-based crack segmentation performance are yet to be systematically investigated.

1.3 Research Objective

The overarching goal of this technical report is to develop novel image-based condition assessment methodologies for civil infrastructures by leveraging advanced sensing and imaging technologies. To achieve this goal, several research objectives aiming at addressing the challenges summarized in section 1.2 by proposing novel image-based crack detection methodologies are stated as below:

i) To develop a robust crack detection methodology by proposing a novel image pre-processing technique with minimal prior user input and subjectivity to effectively eliminate the image disturbances in laser-scanned range image data. By leveraging this methodology, Challenges 2 and 3 which are related with the issues in the image data and associated image pre-processing can be properly addressed. Furthermore, the issue as described by Challenge 4 is also solved through this methodology by using a contour-based crack detection algorithm;

ii) To propose a DCNN-based crack classification methodology and perform a systematic study on the optimal joint hyperparameter configuration on the DCNN architecture and the associated training scheme, providing prior knowledge and insights to future research and applications. Thus, Challenge 5 regarding the hyperparameter selection is tackled;

iii) To propose a DCNN-based crack segmentation methodology which can achieve pixel-level resolution on crack detection and is robust against image disturbances in raw image data under real-world scenarios. This methodology aims to address the issue as described by Challenge 6;

iv) To address Challenges 1, 2, 3, and 7 which are related with the image data, a DCNN-based crack detection methodology is developed to investigate the effect of different types of image data (i.e., heterogeneous image data) on DCNN performance and introduce robustness to DCNN-based applications through heterogeneous image fusion.

Table 1-1 summarizes the proposed methodologies in this technical report and the corresponding challenges addressed by them.

Table 1-1. The proposed methodologies and the corresponding challenges addressed.

Methodologies proposed in this technical report	Section number	Addressed challenges
A robust image processing technique for crack detection using range images	3.1	2, 3, 4
DCNN-based crack classification using range images: A comparative study on hyperparameter selection	3.2	5
DCNN-based crack segmentation using range images	3.3	6
DCNN-based crack classification and segmentation with heterogeneous image fusion	3.4	1, 2, 3, 7

1.4 Layout of the Technical Report

The rest of this technical report is organized in the following manner. Chapter 2 first provides a thorough literature review on image-based crack detection methodologies, and then introduces the technical background related with the proposed DCNN-based methodologies. Chapter 3 describes the proposed image-based crack detection methodologies in detail. Then, in Chapter 4, experimental studies using the proposed methodologies for crack detection on image data of real-world complexities are performed, and the associated results and discussions are presented. Chapter 5 summarizes the findings and conclusions for each proposed methodology, and then provides an overall assessment and conclusion regarding this technical report.

CHAPTER 2: BACKGROUND

Section 2.1 and 2.2 provide a thorough literature review on image-based crack detection methodologies, putting emphasis on introducing recent advancements in DCNN-based crack detection. Then, information on the technical background related with DCNN is briefly described in section 2.3. Section 2.4 introduces the image acquisition system used in this study. And, the last section 2.5 describes the quantitative metrics for performance evaluation.

2.1 Non-Learning-Based Methodologies

Non-learning-based methodologies usually require handcrafted image processing procedures or techniques such as edge detection for feature extraction. Some of the non-learning-based methodologies used intensity image data, relying on the change of pixel intensities in cracked regions to interpret the presence of cracks. Abdel-Qader et al. [46] compared the effectiveness of four edge detection techniques (Canny edge detection, Sobel edge detection, Fourier transform, and fast Haar transform) on crack detection. They concluded that the fast Haar transform had the most accurate detection performance. Salman et al. [32] adopted Gabor filter to detect cracks based on the orientation feature of cracks. Hashimoto [47], Yamaguchi and Hashimoto [37], and Zhu et al. [48] utilized the concept of percolation, a physical model to describe liquid permeation, for crack detection. This approach initializes a crack seed region and labels the neighboring regions as cracks based on the percolation model. Tsai et al. [49] applied an algorithm [50] to detect pavement cracks through probabilistic modeling in crack presence and dynamic programming (DP), and concluded it outperformed other methods in accuracy and robustness. Huang and Xu [51] introduced a real time pavement crack detection algorithm which divides an image into grid cells and identifies crack cells by matching their neighbors with predefined templates/patterns.

In general, because of the intensity-based data acquisition method, the performance of many crack detection algorithms is severely hampered in the presence of shadows, varying lighting conditions, and poor intensity contrast between cracks and surrounding surface [19-23]. Moreover, image noises such as blemishes (e.g., tire marks, oil spills, stains, etc.) [19,24-26] also deteriorate the crack detection performance. To address these issues in intensity images, several pre-processing techniques such as median filtering [52,53] and grayscale multiplication [54] have been adopted to improve the performance of the intensity-based approaches under varying illumination condition. Furthermore, technologies including stereovision and laser imaging were explored by many researchers to obtain three-dimensional (3D) data for crack detection. Wang et al. [25,55-57] performed pioneering study in 3D pavement image processing. They developed and applied an automated pavement distress survey system. In this survey system, intensity image data was collected by a Digital Highway Data Vehicle (DHDV) to reconstruct 3D surfaces through stereovision. Torok, et al. [58] and Jahanshahi and Masri [59] adopted 3D reconstruction based on Structure-from-Motion (SfM) [60] to gain depth perception from intensity images by using multiple cameras. Despite the use of stereovision-based techniques, several inherent issues in the intensity images, especially the varying illumination condition, cannot be completely avoided.

More recently, laser-based 3D data collection system has been adopted, because the laser-scanned data is insensitive to varying lighting condition, and the aforementioned image noises do not interfere with the crack detection by using the collected range (i.e., elevation) image data [19]. Tsai et al. [61,62] developed a road survey system using a Georgia Tech Survey Vehicle (GTSV) mounted with a Laser Crack Measurement System (LCMS [63]), which can collect high-resolution 3D continuous pavement profiles. In a crack detection system proposed by Zhang, et al. [64], 3D pavement surface data was collected by using DHDV with PaveVision3D system. After obtaining the range data, methods such as morphological operation [18], wavelet transform [36], and DP-based method [61,65] were adopted for crack detection.

Methods using range images rely on the difference in elevations between the crack and surrounding regions for crack detection. This type of image data, however, also has issues such as surface variations, scanning noises, and non-crack patterns (e.g., grooves) [7,36]. Similar to intensity-based methods, range-based methods usually employ image pre-processing techniques to tackle these issues prior to crack detection. Ouyang and Xu [36] applied a surface fitting technique to remove the large-scale surface variations (e.g., uneven surface, transverse ripples due to vehicle vibration) in the range images. Jiang and Tsai [65] implemented an outlier removal and profile rectification technique by using a Gaussian filter to flatten the scanned range surface.

Overall, a few challenges and issues as summarized in Challenge 1, 2, 3, and 4 in section 1.2 still exist in the non-learning-based methodology. Especially, the uncertainty and subjectivity due to parameter selection in the image pre-processing and defects detection procedures, as stated in [29,42], is a major challenge for the non-learning-based methods to achieve consistent performance under real-world complexities.

2.2 Learning-Based Methodologies

Recent advancement in computing technologies has paved a way for machine learning-based crack detection methods. This type of methods has drawn researchers' interests by learning from data and making self-adaptations with minimum human intervention.

2.2.1 *Machine Learning*

Some pioneering applications using machine learning for crack pattern recognition were reported in literature. Lee and Lee [66] proposed a crack detection and classification method by integrating an image-based neural network (NN), a histogram-based NN, and a proximity-based NN. Each of them has a 3-layer architecture to detect and classify different crack patterns. Saar and Talvik [67] applied NN to detect cracks and further classify them into separate types including alligator crack, longitudinal crack, and transverse crack. In another machine learning-based application, support vector machine (SVM) was utilized by Moussa and Hussain [68] to classify pavement cracks into alligator crack, block crack, longitudinal crack, and transverse crack. Jahanshahi, et al. [18] evaluated the crack classification performance of three different classifiers: NN, SVM, and nearest neighbor classifier. They demonstrated that the first two classifiers had similar performance and both outperformed the last one. However, the machine learning methods used by these early studies only represent one or two layers of feature abstraction and cannot fully reflect the complexity of typical roadway surfaces [42].

2.2.2 *Deep Learning through DCNN*

As a deep learning-based method, deep convolutional neural network (DCNN) has gained popularity from researchers due to its wide adaptability and versatility. DCNN is advantageous over other image-based methods on that *i*) it can directly learn from the input data, thus resulting in minimum human intervention and prior assumption; and *ii*) it can reflect the data complexity through a hierarchical feature abstraction which is enabled by its multi-level network layout. DCNN architectures are loosely inspired by the biological neural network system in the cerebral cortex, where it processes the input percept using different areas of the cortex to abstract hierarchical levels of features [69]. The term "deep network", opposite to a "shallow network", refers to the fact that a DCNN usually consists of multiple convolutional layers and fully connected layers to extract high-level features from the data.

- Crack classification

Early applications applied DCNNs for crack classification tasks. Zhang, et al. [70] proposed an automated roadway crack detection method, using DCNN to predict whether the input image patches contained cracks or not. They compared it with SVM and the Boosting method, and concluded it had higher classification accuracy. Cha, et al. [39] proposed a DCNN architecture to detect concrete cracks in intensity images under realistic situations such as uneven lighting condition. They showed that it outperformed the traditional edge detection methods. Zhang, et al. [41] applied DCNN to detect pavement cracks and sealed cracks. Park, et al. [71] proposed a roadway crack detection method using DCNN to categorize the image patches into crack, road marking, and intact area.

These crack classification methodologies are patch-based; that is, images are cropped into patches which have smaller sizes, and each image patch is classified by the DCNN as containing cracks or not. Thus, for patch-based methods, resolution on the extracted cracking information is limited.

- Crack object recognition

Another type of task that can be accomplished by DCNN is crack object recognition. Typical applications use region-based DCNNs [72,73] which can generate bounding boxes to enclose cracking features. Cha, et al. [74] modified the faster R-CNN [73] to detect five defective objects including concrete cracks. Maeda, et al. [75] applied different region-based DCNNs to generate bounding boxes for eight different types of roadway defects. Xue and Li [76] proposed a region-based DCNN to detect and locate five types of defects in shield tunnel linings. Nevertheless, the issue of limited resolution on crack detection still exists in some applications using region-based DCNNs.

- Crack segmentation

More recent development in DCNN-based crack detection is pixel-wise crack classification through semantic segmentation. The semantic segmentation is a process to segment an image into different regions by assigning each image pixel with a categorical label. Thus, instead of identifying each image patch as containing cracks or not, semantic DCNNs predict each image pixel as a "crack" or "non-crack" pixel, leading to a higher resolution.

The encoder-decoder network is a type of semantic DCNN which has gained many researchers' interests. The encoder-decoder network contains two essential components: an encoder network to extract features from the input image, and a decoder network to expand the features extracted through the encoder such that the size of the output probability map matches with that of the input image. Yang, et al. [77] proposed an encoder-decoder network based on VGG19 [78] to produce pixel-level crack maps on concrete pavements and walls. Zou, et al. [79] proposed DeepCrack, an encoder-decoder network based on SegNet [80] with multi-scale cross entropy losses. They showed it outperformed some other DCNNs such as U-Net [81] and SegNet on crack segmentation performance. Bang, et al. [14] proposed an encoder-decoder network based on ResNet [82] for roadway crack segmentation on images containing non-road objects. Dung [83] developed an encoder-decoder network with a VGG16-based encoder for crack segmentation on concrete surfaces.

Besides the applications using encoder-decoder networks, some other applications proposed or applied other types of semantic DCNNs for crack segmentation tasks. Zhang, et al. [42], Tong, et al. [30], and Fei, et al. [40] developed CrackNet and its variants, which are different than the encoder-decoder networks in that their extracted feature maps maintain an invariant spatial size throughout all layers. Tan, et al. [84] applied Mask R-CNN [85], which is modified from the faster R-CNN [73] to further improve the detection resolution to pixel level, for crack segmentation.

Overall, it can be observed as a general trend that, in recent decades, image-based crack detection methodologies have gradually evolved from non-learning-based to learning-based, with their detection resolutions improved to pixel level (i.e., crack segmentation). Nevertheless, despite the high adaptability of DCNN-based methodologies, several challenges as stated in Challenge 5, 6, and 7 in section 1.2 remain to be systematically investigated and properly addressed for this type of methodology to achieve robust and accurate detection performance.

2.3 Technical Background for DCNN-Based Methodology

Neural networks, or more precisely Artificial neural networks (ANN), are computing systems developed for performing multi-domain tasks through learning from data [38]. Vaguely inspired by the biological neural network system that constitutes the cerebral cortex, the ANN gains perception from the input data based on hierarchical feature abstraction through neurons and the associated connectivity patterns [38]. An artificial neuron is referred to as the central processing unit of an ANN, each connected to many others through "synapses" to mimic the functionality of biological neurons. A neuron represents a mathematical relationship between the input and output, for example, a weighted sum [38]. As illustrated by an example in Figure 2-1, an ANN often consists of an input layer, multiple hidden layers, and an output layer, where the neurons in each layer are connected with others through "synapses" defined by weights (e.g., W_i and W_j in Figure 2-1). The influence or contribution of the input to each neuron is tuned by adjusting the weights through back-propagation [38]. In Figure 2-1, the forward-inference process refers to the calculation and storage of intermediate variables from the input layer to the output layer; and, the back-propagation process refers to the computation of the gradient given the model loss with respect to the network weights. Through a learning process based on back-propagation, the ANN can address problems such as object recognition by adapting to data reflecting the patterns of a specific domain. More details regarding the network learning process are described in section 2.3.4.

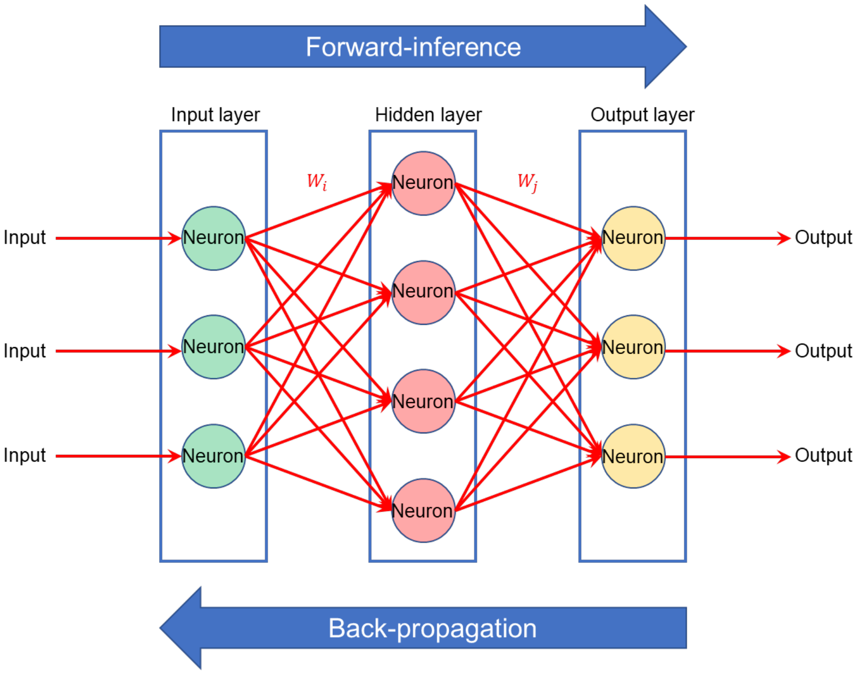


Figure 2-1. A graphical representation of an artificial neural network.

Convolutional neural networks (CNN) [38,86] can be considered as a subclass of ANN, where a CNN takes image data as the input and employs a mathematical operation called convolution [38]. By performing convolution operations on the image data, CNNs can extract spatial features such as edge and shape to facilitate image-based object detection and pattern recognition tasks.

Evolved from CNN, deep convolutional neural network (DCNN) [87] employs multiple (e.g., hundreds of) network layers that constitute a hierarchical layout, as implied by the term "deep" in its name. As described in section 2.2.2, with its high degree of model ability and self-adaptation to real-world scenarios, DCNN can achieve a high-level feature abstraction with minimum human intervention, thus it has become one of the most advanced and prevailing deep learning algorithms in the domain of image-based pattern recognition.

In the following subsections, five topics related with the proposed DCNNs in this study, including the DCNN layers, sliding window technique, data augmentation, training scheme, and residual connection, are introduced.

2.3.1 *DCNN Layers*

In this section, the network layers utilized in this study to construct the proposed DCNN architectures are briefly described. More detailed technical information on these layers can be found in [38,88].

- Convolutional layer

In a DCNN, the functionality of a convolutional layer is to perform feature extraction through a convolution operation. The convolution operation is a dot product between a convolutional kernel and a subregion of the input feature map, plus a bias term, which can be symbolically expressed as Equation (2-1). In this equation, **I** and **Y** are the input and output feature map, respectively; **W** and b are the weights and bias of the convolutional kernel; $\odot$ denotes the dot product. In the convolution operation, the kernel slides horizontally and vertically along the input with a step size called stride. Besides, zero paddings can be added to the input to produce an output with a specific dimension. Figure 2-2 (a) illustrates a two-dimensional (2D) example of the convolution operation through a matrix representation. In this example, a 3×3 (i.e., height×width, same hereafter) convolutional kernel is applied to a 4×4 input feature map with a stride of 1, resulting in a 2×2 output. In Figure 2-2 (a), W_{ij}, I_{ij}, and Y_{ij} denote the elemental entry of the weight matrix **W**, input matrix **I**, and output matrix **Y**, respectively; b denotes the kernel bias scalar; **J** denotes a matrix of all ones whose dimension is the same as **Y**. It can be seen that the convolution operation can be concisely represented as a matrix multiplication. The transposed convolution is also illustrated in Figure 2-2 (b), which can be interpreted as the inverse process of the convolution operation. The transposed convolution is described in section 2.3.1. It is worth noting that in this figure, to keep a concise narrative, the same symbolic notations are used to explain the convolution and transposed convolution operation, but they carry different values.

$$\mathbf{Y} = \mathbf{W} \odot \mathbf{I} + b \cdot \mathbf{J} \tag{2-1}$$

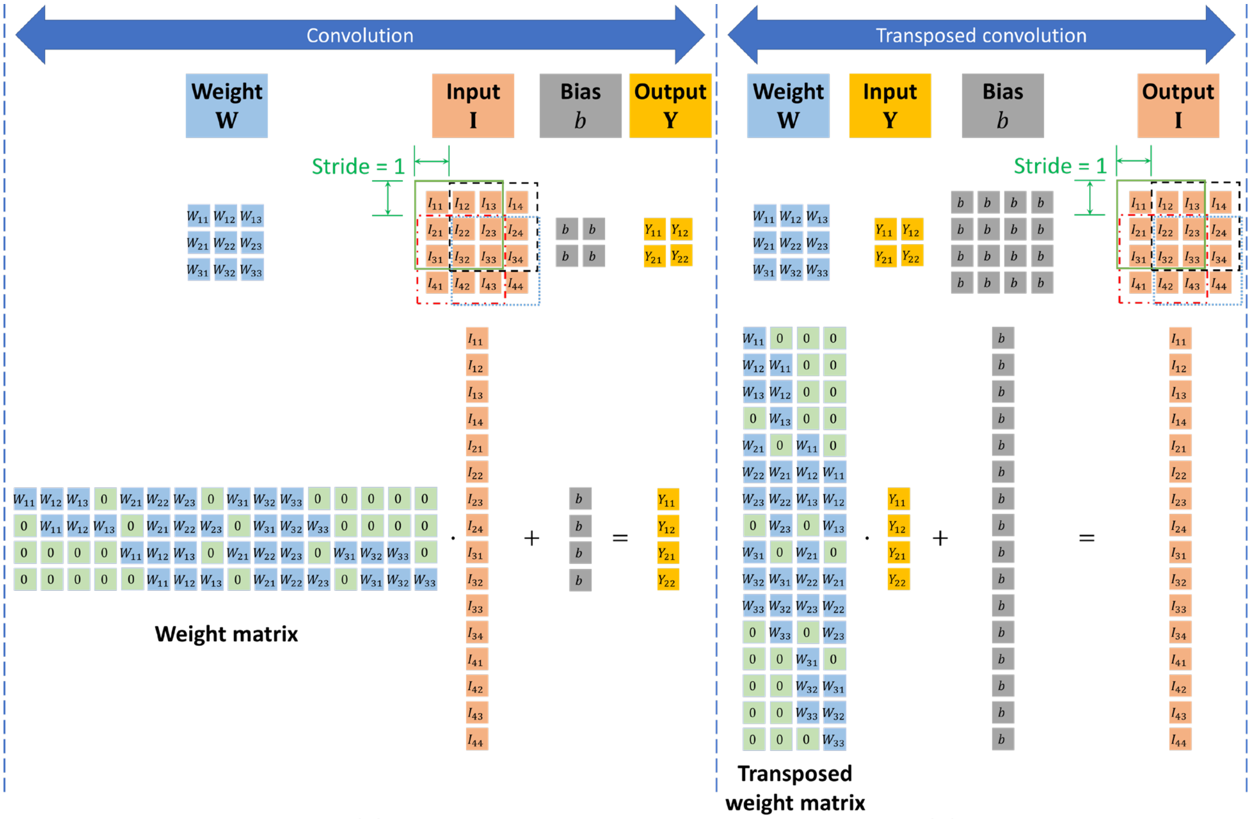


Figure 2-2. A matrix representation of the convolution and transposed convolution operation: (a) convolution; and (b) transposed convolution.

- Transposed convolutional layer

As implied by the term “transposed”, the sparse matrix representation of the kernel weights in the transposed convolutional layer is symbolically the transpose of the weight matrix in the corresponding convolutional layer, as illustrated in Figure 2-2. Similar with the use of zero paddings in the convolution operation, boundary cropping is also often used in the transposed convolution to trim the output size. Figure 2-2 (b) shows a 2D example of the transposed convolution operation, in which a 2×2 input feature map is up-sampled into a 4×4 output by using a 3×3 transposed convolutional kernel with a stride of 1. In a semantic DCNN, multiple transposed convolutional layers with a stride larger than 1 are usually employed to gradually expand the size of the feature map. It is noted that each down-sampling process through either a convolutional or max pooling layer is paired with an up-sampling process performed by a transposed convolutional layer, thus the DCNN is able to maintain the same dimension (i.e., height and width) between the input image and the output probability map.

- Max pooling layer

In DCNN applications, it is often needed to reduce the dimension of the input representation such as the input image or hidden-layer feature map. By performing dimensionality reduction, the computational cost is reduced accordingly, thus improving DCNN training efficiency. Max pooling is an effective process often used in DCNN classification or segmentation tasks for dimensionality reduction [38]. A max pooling layer performs down-sampling on the input feature map by only keeping the maximum value in each kernel, as illustrated in Figure 2-3. In this figure, a 2×2 max pooling kernel with a stride of 2 is applied to the input feature map which has the dimension as 4×4. Correspondingly, the output reserves the maximum value in each subregion, resulting in a smaller size as 2×2. Through the max pooling operation, the feature map is down-scaled by a factor of 2. The difference in terms of down-sampling by the max pooling vs. the convolution operation is that no learning process is involved in the max pooling layer.

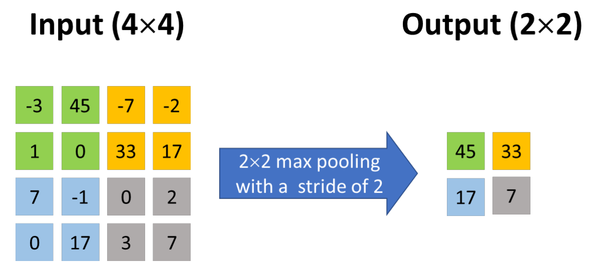


Figure 2-3. An example of the max pooling operation.

- Batch normalization layer

Batch normalization is a regularization technique proposed by [89] to address the issue of internal covariate shift. This issue often occurs during training, when the distribution of inputs to each layer shifts due to the change in network parameters [89]. Such an internal covariate shift is undesirable, because it potentially shifts the original training problem and thus deteriorate the model performance. The batch normalization algorithm employs two learnable parameters called scale and shift factors to normalize the input distribution in each batch, such that the internal covariate shift issue is mitigated. The formulation of batch normalization is provided in Figure 2-4 [89]. As shown in this figure, first, an activation x over a mini-batch is normalized by its mean and variance; then, the distribution of the normalized values of x is modified by the two learnable parameters as the scale γ and shift β.

```
Input:  Values of $x$ over a mini-batch: $B = \{x_1, \dots, x_m\}$;
        Parameters to be learned: $\gamma$, $\beta$

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

        $\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$                          (mini-batch mean)

        $\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$            (mini-batch variance)

        $\widehat{x_i} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$                    (normalize)

        $y_i \leftarrow \gamma\widehat{x_i} + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$    (scale and shift)
```

Figure 2-4. The batch normalization algorithm [89].

- Nonlinear activation layer

The leaky rectified linear unit (LReLU) [90] is adopted as a nonlinear activation function to add nonlinearity to a DCNN model. The formulation of LReLU is provided in Equation (2-2). As expressed in Equation (2-2), LReLU is a bi-linear function, where the gradient is equal to 1 for the non-negative neuron input, and a small positive value α for the negative input. By activating the negative neuron inputs during back-propagation with a small positive gradient α, LReLU can avoid the "dying" neuron problem [38,90] which often occurs upon using the rectified linear unit (ReLU) [91].

$$\mathrm{LReLU}\ (x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases} \tag{2-2}$$

where x is the neuron input; α is a small positive value.

- Dropout layer

Dropout was proposed by [92] as an effective regularization technique to address the issue of overfitting. Overfitting occurs when a DCNN becomes too attuned to the training dataset that it loses the ability to properly adapt to a new dataset. The dropout layer prevents overfitting by randomly deactivating neurons from the previous layer with a certain probability (e.g., 50%). During the training process, dropout forces the network to adapt to different neurons, thus improving generalization.

- Fully connected layer

A fully connected layer is usually adopted in a DCNN for image classification tasks. The term "fully connected" implies that each neuron in this layer has connections to all the neurons in the previous layer, as described by Equation (2-3). This operation is a matrix multiplication plus a bias vector. The fully connected layer can largely reduce the size of the feature map, and the output (i.e., scores) from this layer is usually mapped to a probability distribution for class prediction.

$$\mathbf{y} = \mathbf{W} \cdot \mathbf{x} + \mathbf{b} \tag{2-3}$$

in this equation, **x** denotes a vector flattened from the neuron input; **W** denotes the weight matrix of the fully connected layer; **b** denotes the bias vector; and, **y** denotes the neuron output.

- Softmax layer

A softmax layer is usually placed at the end of a DCNN to normalize the pixel values into a probability of each pixel belonging to a specific class. The softmax function is expressed as Equation (2-4). In this equation, x_i denotes the pixel value at the i^{th} depth channel of the input; n denotes the number of classes which is equal to the depth (i.e., third dimension) of the input matrix. Softmax (x_i) produces the probability of each pixel belonging to the i^{th} class.

$$\text{Softmax } (x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{2-4}$$

2.3.2 *Sliding Window Technique*

The sliding window technique is adopted by many researchers [39,93,94] to reduce the dimension of input image and increase the number of image samples through a cropping operation. As illustrated by Figure 2-5 (a), a square kernel slides along the horizontal and vertical direction of an image, with 50% overlap, to crop the original image into image patches which have smaller spatial dimensions to reduce computational cost. In this study, image patches of 256×256 pixels are generated for training and testing.

Figure 2-5 (b) shows the zoomed-in view of the sliding window process, where different data regions are utilized multiple times during prediction. Then, a summed crack probability map with the same size as the original image is generated through arithmetic addition, which is an inverse operation of the sliding window process; the values in the summed probability map are divided by the divisors as shown in Figure 2-5 (b), based on how many times each image pixel is predicted. Finally, a binarized crack map is generated from the crack probability map with a threshold of 0.5, where white pixels indicate cracks and black ones indicate the background.

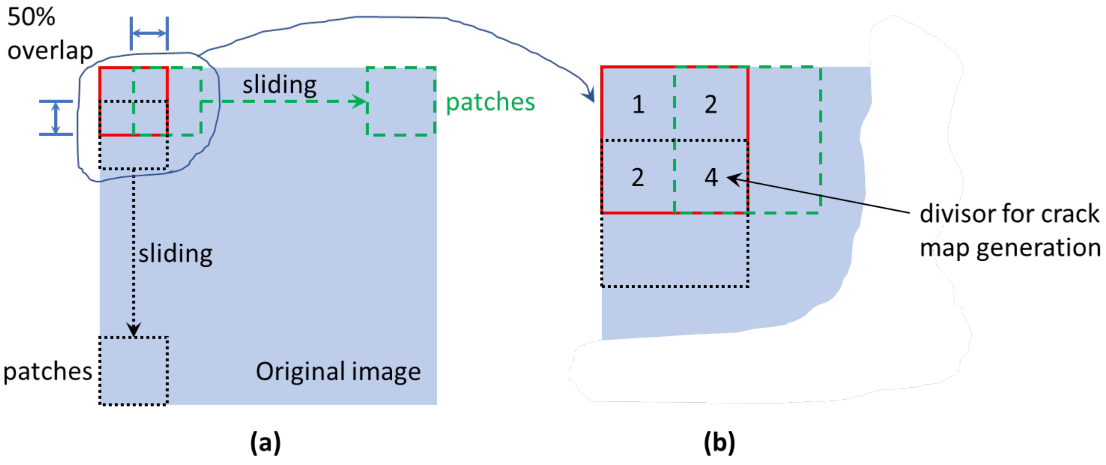


Figure 2-5. The sliding window technique: (a) image patch generation; and (b) zoomed-in view.

2.3.3 *Data Augmentation*

Data augmentation is another effective means to reduce overfitting and improve generalization of a DCNN by artificially increasing the number of image data [95]. It is a process to generate additional samples of an image by applying label-preserving transformations such as rotation, which leave the underlying class unchanged [96]. Common data augmentation techniques include random rotation, translation, mirroring, and scaling, etc. [97]. In this study, the number of image patches is effectively increased through these data augmentation techniques.

2.3.4 *Training*

Three topics related with the DCNN training process, including the cost function, optimization algorithm, and parameter initialization, are introduced in this section.

- Cost function

In this study, the cross entropy loss [98] is employed as the cost function to measure the discrepancy between the ground truth and prediction. The formulation of the cross entropy loss is expressed in Equation (2-5).

$$E = -\frac{1}{m}\sum_{k=1}^{m}\sum_{j=1}^{N}\sum_{i=1}^{n}\left[t_{j,i}^{k} \cdot \ln\left(p_{j,i}^{k}\right)\right] + \Omega(\mathbf{w}) \qquad \textbf{(2-5a)}$$

$$\Omega(\mathbf{w}) = \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} \qquad \textbf{(2-5b)}$$

in this equation, E denotes the cross entropy loss; m denotes the mini-batch size, which refers to the number of input images participating in the training in each iteration; $t_{j,i}^{k}$ is a logical operator, where the superscript k denotes the k^{th} image; the subscripts j and i indicate the logical operator is evaluated at the j^{th} pixel for the i^{th} class; N is the total number of pixels in each image; $t_{j,i}^{k}$ is equal to 1 if the pixel belongs to the i^{th} class and 0 otherwise; n is the number of classes, which is equal to 2 (i.e., "crack" vs. "non-crack") in this study; $p_{j,i}^{k}$ is the probability of the pixel belonging to the i^{th} class, which is calculated by Equation (2-4). An additional penalty term expressed as Equation (2-5b) is introduced in Equation (2-5a) to regularize the training process by penalizing large weights. In Equation (2-5b), $\mathbf{w}$ denotes a vector whose entries are the learnable parameters including the weights in the convolutional layers, fully connected layers, transposed convolutional layers, and batch normalization layers (i.e., scale and shift factors); the superscript T denotes the transpose operation; λ is a hyperparameter called weight decay factor to adjust the degree of penalization to the weights.

- Optimization algorithm

During DCNN training, the objective is to reduce the discrepancy between the ground truth and prediction by minimizing the cost function. In this study, the mini-batch stochastic gradient descent (SGD) with momentum [99] is adopted as the optimization algorithm. The formulation is expressed in Equation (2-6). In this equation, $\boldsymbol{\theta}$ is the model parameter vector which consists of two components: the weight vector $\mathbf{w}$ and bias vector $\mathbf{b}$ in the DCNN; $\boldsymbol{\theta}_t, m_t$ with a subscript t indicate the parameter vector and mini-batch input at the t^{th} iteration, respectively; E denotes the cross entropy loss, calculated by Equation (2-5); ε is a hyperparameter called momentum which adjusts the learning speed; r is another hyperparameter to tune the efficiency of learning which is referred to as the learning rate.

i) Learning scheme

$$\Delta\boldsymbol{\theta}_{t+1} = \varepsilon \cdot \boldsymbol{\theta}_t - r \cdot \left.\frac{\partial E}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}_t, m_t} \tag{2-6a}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \Delta\boldsymbol{\theta}_{t+1} \tag{2-6b}$$

$$\text{from epoch } 2(M-1) \text{ to } 2M,\, r = (0.8)^{M-1} \cdot r_{ini},\, M \in \mathbb{Z}^{+} \tag{2-6c}$$

as expressed in Equation (2-6c), a piece-wise learning rate scheme is employed for the optimization in this study; that is, the learning rate decays every 2 epochs by a drop factor of 0.8. An epoch is a full pass of all the training data through mini-batches. In Equation (2-6c), r_{ini} denotes the initial learning rate. A step-wise decayed learning rate can encourage fast convergence at the early stage of training and enable fine-tuning on the model parameters at the late stage.

- Parameter initialization

i) Weight and bias

In this study, upon training DCNNs, the weights associated with the convolutional layers, fully connected layers, and transposed convolutional layers are initialized by the Glorot initializer [100].

This initializer independently samples each weight from a Gaussian distribution with zero mean and a variance which is related with the dimension of the weights. The initial biases are set as zero.

ii) Scale and shift factor

The scale and shift factor in each batch normalization layer are also updated during training. They are initialized as 1 and 0, respectively.

2.3.5 *Residual Connection*

In DCNN-related studies, an intuitive strategy to improve the network performance is to increase the model capacity by stacking more layers. However, as discovered by some researchers [82,101,102], increasing the network depth does not always result in improved performance; the deeper architecture might not outperform its shallower counterpart due to performance degradation brought by considerably increased network depth. To alleviate such issues in deep architectures, the concept of residual connection (also known as skip connection) was proposed by [82], as illustrated by Figure 2-6. The residual connection can improve the network performance by integrating the hierarchical features from different layers through a shortcut connection, which facilitates the training process [82]. Also, according to [101], the use of residual connections in deep architectures can alleviate the issues of singularities in models which slow down the learning process and lead to performance degradation.

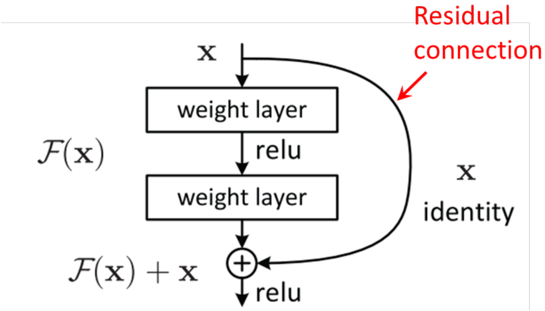

Residual connection
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2-6. Residual connection (redrawn from [82]).

2.4 Laser Image and Measurement System

A laser imaging system, manufactured by AMES Engineering, is adopted to capture both range (i.e., elevation) and intensity image data from roadway surfaces. Integrated into a survey vehicle, this laser imaging system consists of three components: a 3D laser camera [Figure 2-7 (a)], a data acquisition (DAQ) module [Figure 2-7 (b)], and an on-board data processing computer [Figure 2-7 (c)]. The 3D camera is mounted on the rear top of the vehicle. The vertical distance between the camera module and the ground is 2.13 m (84 inch), such that the camera can capture a 3.96 m (156 inch) wide transverse profile during each scan. The scanned profile contains 4096 pixels, making the transverse pixel resolution as close to 1 mm/pixel (0.04 inch/pixel). Along the longitudinal direction, to maintain a uniform longitudinal pixel resolution as 2 mm/pixel (0.08 inch/pixel), the sampling rate of the DAQ is set at 4856 Hz when the vehicle is driving under 9.83 m/s (22 mph). The depth resolution of the acquired range image data is 0.1 mm.

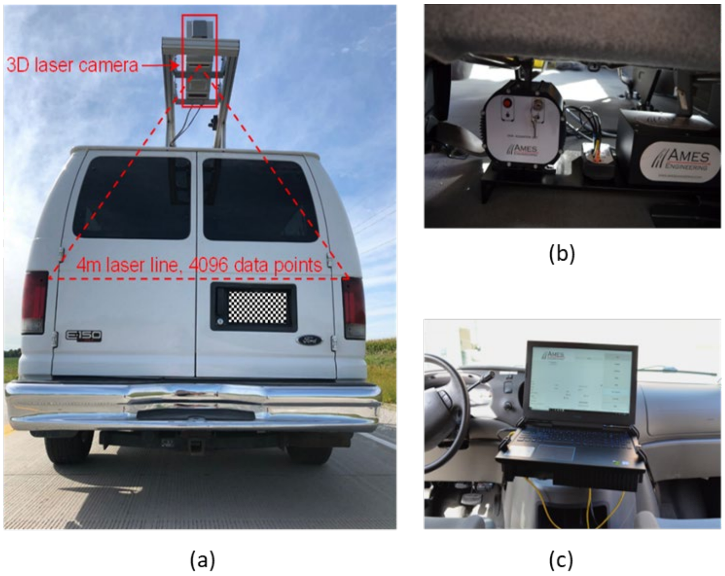


Figure 2-7. The laser imaging system: (a) vehicle-mounted 3D camera; (b) DAQ module; and (c) data processing computer.

2.4.1 *Measuring Principle*

The range and intensity image data are obtained through laser-based 3D triangulation, as illustrated by Figure 2-8. The laser line projector is positioned perpendicular to the measurement plane, while the laser camera views the object from an angle. During each scan, a laser line is projected perpendicular to the roadway surface. By measuring the intensity value at each pixel location, an intensity profile can be captured. Meanwhile, by interpreting the pixel shifts in the detector view, as shown in Figure 2-8, the range information at each pixel location can also be calculated through 3D triangulation. A range profile at the same location where the laser line is projected can be obtained along with the intensity profile. Thus, the acquired range and intensity image data in this study have pixel-to-pixel location correspondence.

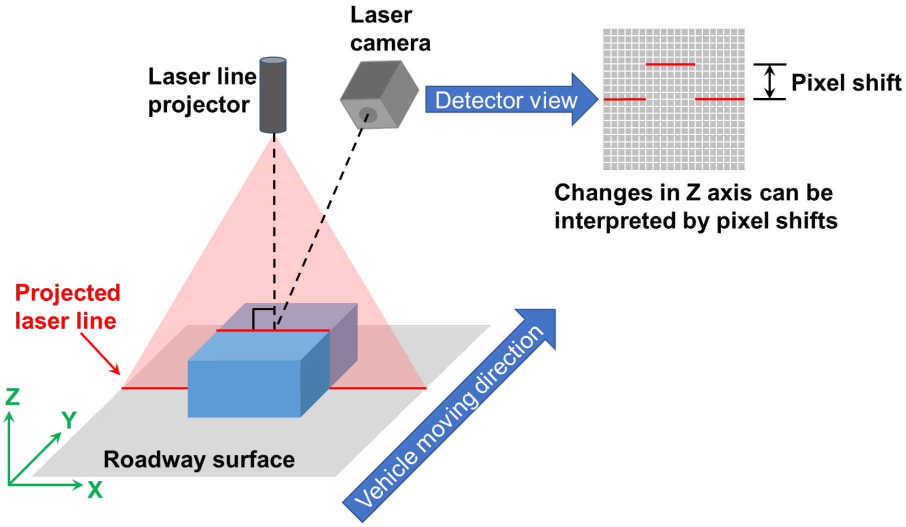


Figure 2-8. Laser-based 3D triangulation.

2.5 Performance Evaluation

2.5.1 *Precision-Recall Analysis*

In binary segmentation study, the number of foreground objects is usually much fewer than that of background, which is referred to as the class imbalance issue [103]. In the case of crack segmentation, the ratio between the total number of foreground (i.e., crack) and background (i.e., non-crack) pixels is usually quite small. One of the most straightforward and commonly used metrics, Accuracy, has very poor performance on a class-imbalanced dataset. Accuracy is defined as the number of correctly identified pixels over all predicted pixels. Given an image where the majority belongs to the non-crack class, for example, 90%, the Accuracy value will still be as high as 90% even if all the pixels are predicted as non-crack ones and none of the true crack pixels is correctly identified. Therefore, the adopted metrics need to be able to address the issue due to class imbalance.

In this study, the precision-recall analysis [104] is adopted to evaluate the performance of the proposed crack segmentation methodologies on a class-imbalanced dataset. Three metrics including the Precision, Recall, and F1 score are included in this analysis, as expressed in Equation (2-7). In this equation, Precision is defined as the ratio of the number of correctly identified true crack pixels to the number of pixels predicted as cracks; Recall is defined as the ratio of the number of correctly identified true crack pixels to the number of true crack pixels; and, F1 score is the harmonic mean of Precision and Recall, which provides a comprehensive measure on the segmentation performance.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2-7a}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2-7b}$$

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2-7c}$$

where TP denotes the number of correctly identified true crack pixels; FP denotes the number of non-crack pixels that are misidentified as crack pixels; and, FN denotes the number of true crack pixels misidentified as non-crack pixels. TP, FP, FN, and TN (i.e., true negative) are illustrated in Figure 2-9.

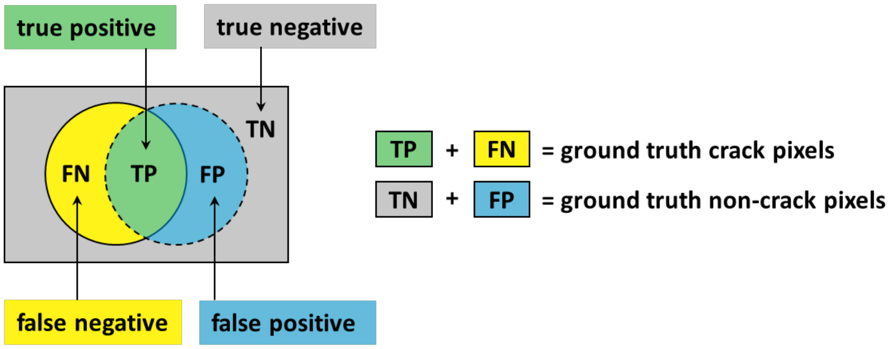

true positive
true negative
TN
FN
TP
FP
false negative
false positive
TP + FN = ground truth crack pixels
TN + FP = ground truth non-crack pixels

Figure 2-9. A schematic diagram to explain the performance metrics.

2.5.2 *Intersection over Union*

Intersection over union (IOU) [105], also known as the Jaccard index, is commonly used for performance evaluation in the field of image segmentation. The expression of IOU is formulated in Equation (2-8), in which the definition of TP, FP, and TN is the same as defined in Equation (2-7). As illustrated in Figure 2-9, IOU reflects the degree of overlap between two objects. In this study, the IOU is evaluated on the "crack" class to provide a measure on the overlap between the ground truth crack object and predicted crack object.

$$\mathrm{IOU} = \frac{TP}{TP + FP + FN} \tag{2-8}$$

2.5.3 *Boundary F1 Score*

Other than F1 and IOU, another type of metrics called boundary F1 (BF) score [106] is also adopted in this study. As implied by the name, the BF score is the F1 score extended to semantic segmentation, which quantitatively measures the similarity between the boundary (i.e., a contour) of the ground truth object and that of the predicted object. The BF score varies from 0 to 1, where the value 1 indicates an exact match between the contours of two objects. The formulation of the BF score is provided in [106]. In this study, the BF score is calculated on the "crack" class to offer an additional measure on the similarity between the ground truth of crack and the predicted crack object.

CHAPTER 3: PROPOSED TECHNIQUES

This chapter describes the proposed image-based crack detection techniques to address the challenges mentioned in section 1.2. The detailed organization is as follows: *i*) a non-learning-based technique by using frequency domain filtering is proposed in section 3.1 to address the issues existing in range image data; *ii*) in section 3.2, a series of DCNN classifiers with different hyperparameter configurations are developed, to investigate the impact from hyperparameter selection on roadway crack classification with range image data; *iii*) section 3.3 introduces the proposed DCNN-based crack segmentation technique to achieve pixel-level detection performance on range image data; and *iv*) in section 3.4, deep learning-based data fusion techniques for both crack classification and segmentation are proposed to tackle image data of real-world complexities and achieve robust detection performance.

3.1 Image Processing Technique for Robust Crack Detection Using Range Image Data

3.1.1 *Motivation*

One issue with most of the non-learning-based crack detection methods (section 2.1) is that they usually do not consider crack connectivity and boundary [37], and that they have the drawbacks as being sensitive to image noises [107]. It is difficult to extract the cracking properties such as width and area if enclosed crack boundaries are not identified. Instead, contouring analysis can provide an enclosed boundary for each individual crack identified and offer accurate estimate on the cracking properties. Contouring analysis is an imaging technique which has been widely used in geology and oceanology to interpret terrain information [108-111], but it has not been explored for crack detection. In this study, a boundary extraction algorithm (i.e., marching squares) for contouring analysis is employed to extract single-level contour candidates for the subsequent crack detection and classification process. The assumption of applying this image technique for crack detection is that, cracked regions have different elevations than the surrounding areas. Based on this assumption, one necessary step before applying the contouring analysis is to ensure the analyzed image surface is free from *i*) non-crack surface variations [7], which are usually caused by rutting [Figure 3-1 (a)], uneven lanes [Figure 3-1 (a)], and ripples due to vertical vibration of the data acquisition vehicle [Figure 3-1 (b)] [36]; and *ii*) grooved patterns [Figure 3-1 (c)]. If these features exist in the data, the application of contouring analysis might lead to misidentification on surface cracks.

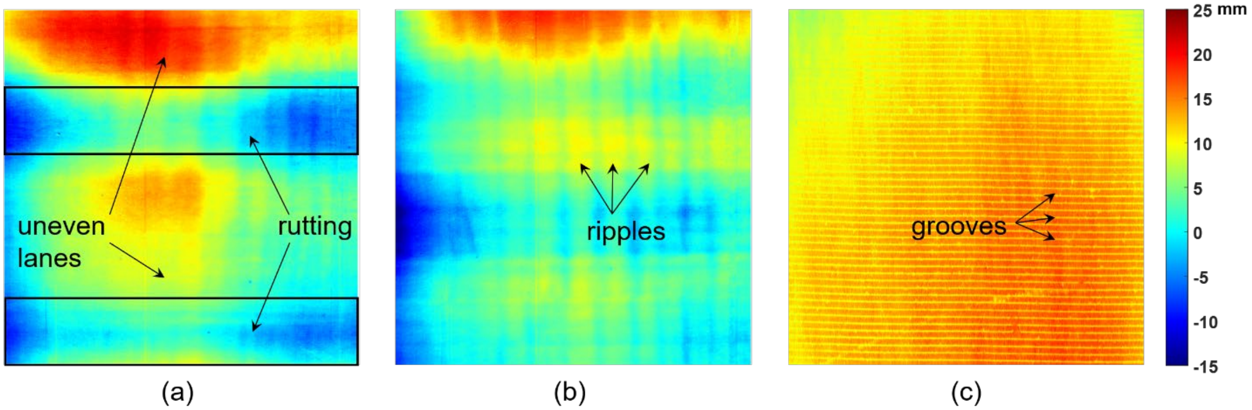


Figure 3-1. Surface variations: (a) rutting and uneven lanes; (b) ripples due to vehicle vibration; and (c) grooved patterns.

Another issue is related with parameter selection in image pre-processing. To facilitate a robust crack detection via contouring analysis, critical pre-processing steps are required to remove the surface variations and grooved patterns, which are usually characterized by their unique features in frequency domain comparing against those of cracks. Existing spatial domain methods, including histogram transformation [54,112,113], median-filter-based crack enhancement [52,53,114], and surface fitting [36], are applied for background correction and crack enhancement purposes, but they often require subjective prior knowledge on parameter selection; meanwhile, very few studies [32,115] have explored the idea of using frequency domain filtering techniques on crack detection and removal of surface variations. And yet, rigorous justification in the selection of the filter parameters is still missing in these studies.

To address these issues, a robust crack detection methodology that utilizes frequency domain filtering and contouring analysis is proposed. This methodology makes use of range image data to accurately detect cracks based on their elevation information under realistic field conditions, including image noises, surface variations, and grooved patterns. More specifically, during image pre-processing of the original range data, frequency domain filtering is employed to remove surface variations and grooved patterns and suppress image noises. Unlike most of the non-learning-based methods which are usually involved with subjective parameter selection, in the proposed technique, determination on the parameters (e.g., cutoff frequency) is based on a physical relationship between the crack width and its frequency domain information derived herein. Then, the contouring analysis is applied to extract cracks from the filtered range surface with enclosed boundaries. Thus, by proposing this methodology, Challenges 2, 3, and 4 as described in section 1.2, which correspond to the issues of disturbances in range image data, subjectivities from image pre-processing, and crack boundaries, respectively, can be effectively addressed.

3.1.2 *Proposed Methodology*

- Flow chart of the proposed crack detection methodology

The flow chart of this methodology is illustrated in Figure 3-2. In the flow chart, blue blocks with a dashed border represent the existing procedures, and green blocks with a solid border represent the novel procedures developed in the proposed methodology. This methodology first implements the frequency domain filtering based on a derived relationship between the crack width and cutoff frequency to remove the inherent issues in the range images including surface variations, image noises, and grooved patterns; then, the marching squares algorithm is employed to detect single-level contours thresholded by the crack depth; finally, a set of logics and criteria is developed for contour qualification and crack classification. Future research effort will be devoted to extracting cracking features such as crack width from the detected crack contours. The novelty of this methodology is that it provides a systematic framework to remove the inherent issues in the range images with less-subjective parameter selection, and that it develops a set of logics and criteria for effective crack classification on contour-based crack detection.

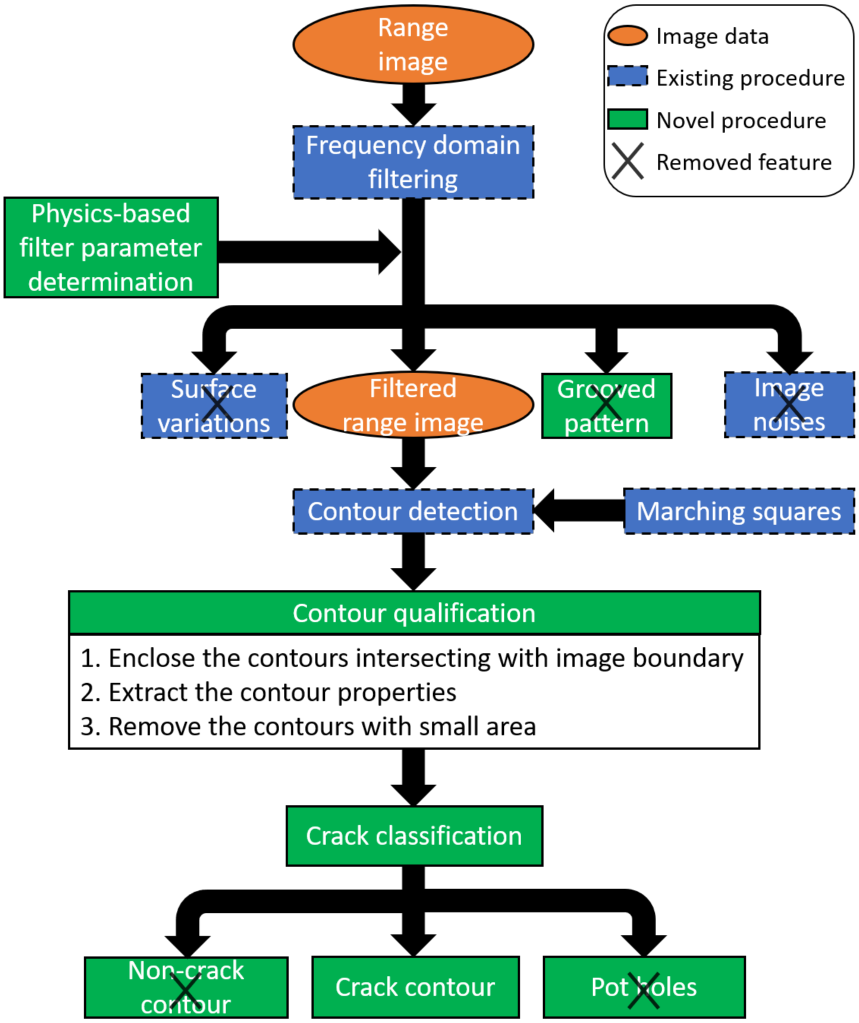


Figure 3-2. Flow chart of the proposed crack detection methodology.

3.1.3 *3D Laser Range Image Data*

The proposed crack detection methodology makes use of range image data. The advantages of using range data over intensity data reside in two aspects: *i*) when adopting range image for surface crack detection, cracks are physically associated with the elevation of the surface, where the cracked regions typically have different elevations than their surrounding areas; and *ii*) using range data can reduce the influences from varying illumination condition, blemishes, and low contrast between cracks and surrounding surface. Figure 3-3 illustrates an acquired bridge deck image data [Figure 3-3 (a)] with its close-up details [Figure 3-3 (b) and(c)], which is captured by the laser imaging system as introduced in section 2.4.

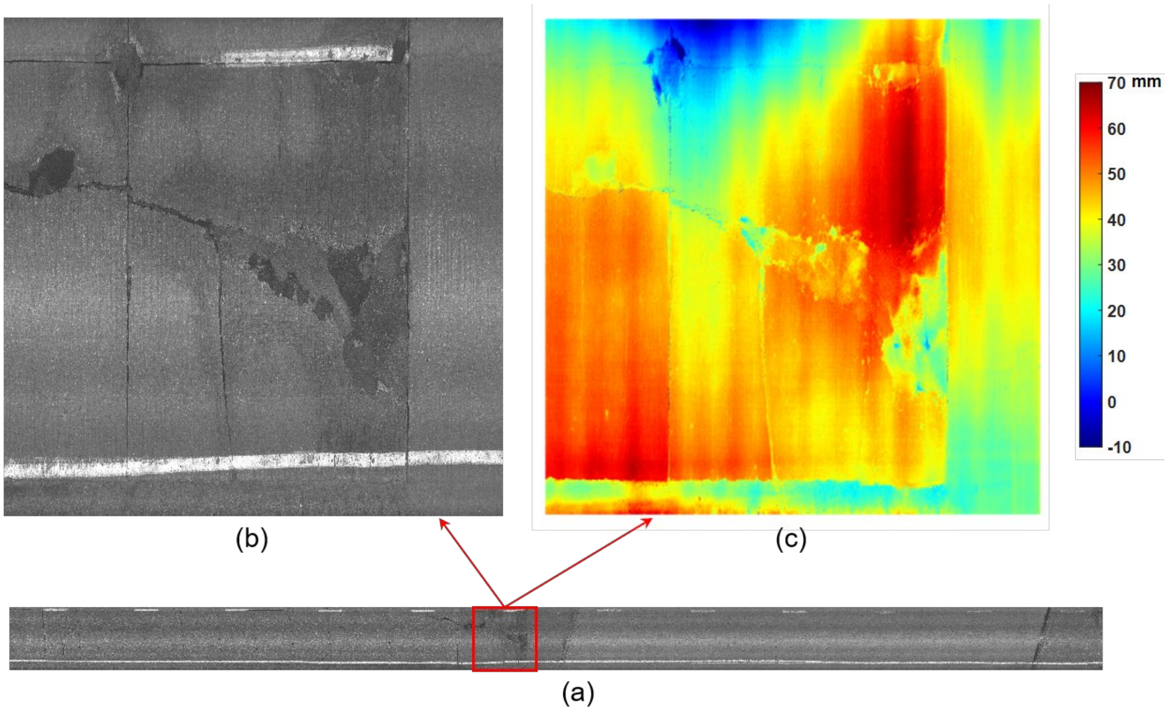


Figure 3-3. A bridge deck image data with close-up details: (a) bridge deck intensity data; (b) intensity image; and (c) range image.

3.1.4 *Frequency Domain Filtering*

In the pre-processing stage, frequency domain filtering techniques are applied to remove surface variations and suppress image noises. In a digital image, crack edges and image noises mostly contain high frequency contents; In contrast, surface variations due to rutting, uneven lanes, and vertical vibration of the data acquisition vehicle (illustrated as Figure 3-1) contain low frequency contents. The goal of applying frequency domain filtering is, by designing filters with appropriate parameter settings, the unwanted surface features in crack detection (i.e., noises and surface variations) can be eliminated effectively; meanwhile, the critical cracking information such as crack edges is preserved during the filtering process. The Discrete Fourier Transform (DFT) is applied to the range image data z_0, to acquire the 2D DFT coefficients Z_0; e.g., the 2D DFT of an M-by-N (pixels) image can be formulated as Equation (3-1). Then, a comprehensive filtering process is conducted on the obtained DFT coefficients Z_0 to obtain filtered coefficients Z_1. Finally, the inverse DFT [formulated as Equation (3-2)] is performed on the filtered coefficients Z_1, to reconstruct the filtered image surface z_1.

$$Z_0(k,l) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} z_0(x,y) e^{-2\pi i (\frac{kx}{M} + \frac{ly}{N})} \quad \textbf{(3-1)}$$

$$z_1(x,y) = \frac{1}{\sqrt{MN}} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} Z_1(k,l) e^{2\pi i (\frac{kx}{M} + \frac{ly}{N})} \quad \textbf{(3-2)}$$

where $z(x,y)$ is the original range data before filtering, with the pixel coordinates x and y ranging as $0 \leq x \leq M-1$ and $0 \leq y \leq N-1$; $Z(k,l)$ is the DFT coefficient, where k,l are the row and column indices with $0 \leq k \leq M-1$ and $0 \leq l \leq N-1$. The subscripts "0" and "1" indicate the status as before and after filtering process, respectively.

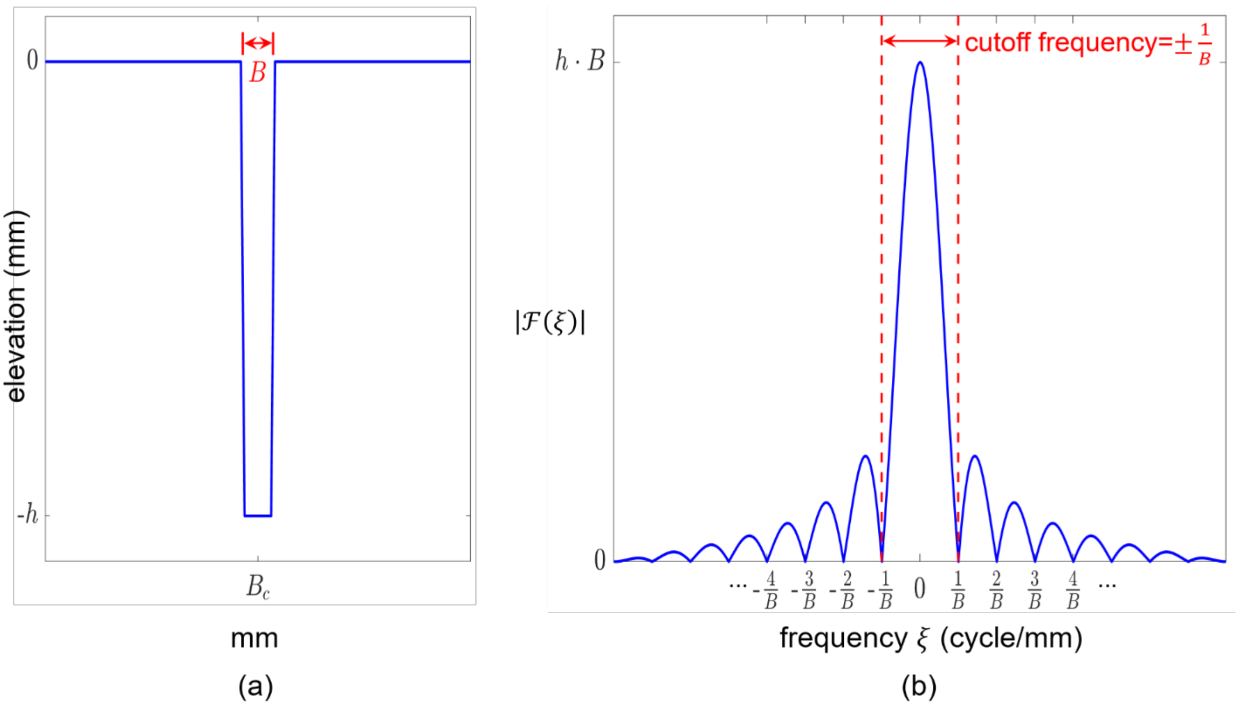


Figure 3-4. Illustration of a simulated crack: (a) spatial domain; and (b) frequency domain.

- Relationship between crack width and cutoff frequency

To determine the parameters for the frequency filtering process, a physical relationship between the width of a crack and its frequency domain information is derived herein. A rectangular-shaped crack is simulated in a flat surface with zero elevation, as illustrated in Figure 3-4 (a); the mathematical expression of the crack is formulated as Equation (3-3a), where B and h denote the width and depth of the crack, respectively. Both B and h are in millimeter (mm). It can be shown that, the Fourier transform of Equation (3-3a) is expressed as Equation (3-3b). Subsequently, the magnitude of the Fourier transform is formulated as Equation (3-3c), which is illustrated in Figure 3-4 (b). By letting Equation (3-3c) equal to zero [see Equation (3-3d)], the frequency bandwidth within $\pm\frac{1}{B}$ cycle/mm (i.e., the main lobe) is obtained as the bandwidth where most of the image energy is located. This frequency bandwidth is indicated by the range constrained within the red dashed lines in Figure 3-4 (b). Therefore, when designing a low-pass filter to suppress image noise whilst preserving this crack, the corresponding cutoff frequency can be set as $\frac{1}{B}$ cycle/mm, i.e., inversely proportional to the crack width B. This relationship illustrates the cutoff frequency can be physically associated with the crack width. By assigning such a cutoff frequency, majority of the image energy associated with the crack shape, which concentrates at the main lobe, resides in the frequency pass band. Likewise, based on this relationship, the maximum and minimum width of the cracks preserved through the filtering can be achieved by properly assigning the cutoff frequencies of the high-pass and low-pass filter, respectively.

$$f(x) = -h \cdot \mathrm{rect}\left(\frac{x - B_c}{B}\right) \tag{3-3a}$$

$$\mathcal{F}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi\xi x} dx = -hB \cdot \mathrm{sinc}(\xi B) \cdot e^{-j2\pi\xi B_c} \tag{3-3b}$$

$$|\mathcal{F}(\xi)| = hB \cdot |\mathrm{sinc}(\xi B)| = hB \cdot \left|\frac{\sin(\pi\xi B)}{\pi\xi B}\right| \tag{3-3c}$$

$$|\mathcal{F}(\xi)| = 0, \text{at } \xi = \pm\frac{i}{B}, i \in \mathbb{Z}^{+} \tag{3-3d}$$

in Equation (3-3a), $f(x)$ denotes the rectangular-shaped crack function; h is the crack depth; x is the coordinate in length; B_c is the horizontal offset from the origin, and B denotes the crack width; h, x, B_c, and B carry the same physical unit as mm; $\mathrm{rect}(\cdot)$ is the rectangular function, defined in Equation (3-4a). In Equation (3-3b), $\mathcal{F}(\xi)$ denotes the Fourier transform, where ξ is the frequency; $\mathrm{sinc}(\cdot)$ is the normalized sinc function, defined in Equation (3-4b).

$$\mathrm{rect}(t) = \begin{cases} 0 & if\ |t| > \frac{1}{2} \\ \frac{1}{2} & if\ |t| = \frac{1}{2} \\ 1 & if\ |t| < \frac{1}{2} \end{cases} \tag{3-4a}$$

$$\mathrm{sinc}(t) = \frac{\sin(\pi t)}{\pi t} \tag{3-4b}$$

- Choice of filters

On the choice of filters for the proposed frequency domain filtering, three types of image filters including the Ideal filter, the Butterworth filter, and the Gaussian filter [116] have been considered. The Ideal filter has a sharp transition between the pass band and stop band, producing ripples known as the ringing artifact [116] near the sharp edges of the filtered spatial domain image. Such ringing artifact is often undesirable and will cause detrimental effects on the subsequent crack detection. The Butterworth filter has a smoother transition band compared with the Ideal filter, which helps reduce the ringing artifact. The Gaussian filter has the identical non-oscillatory form in both the spatial and frequency domain [117], thus does not incur any ringing artifact in the filtered image.

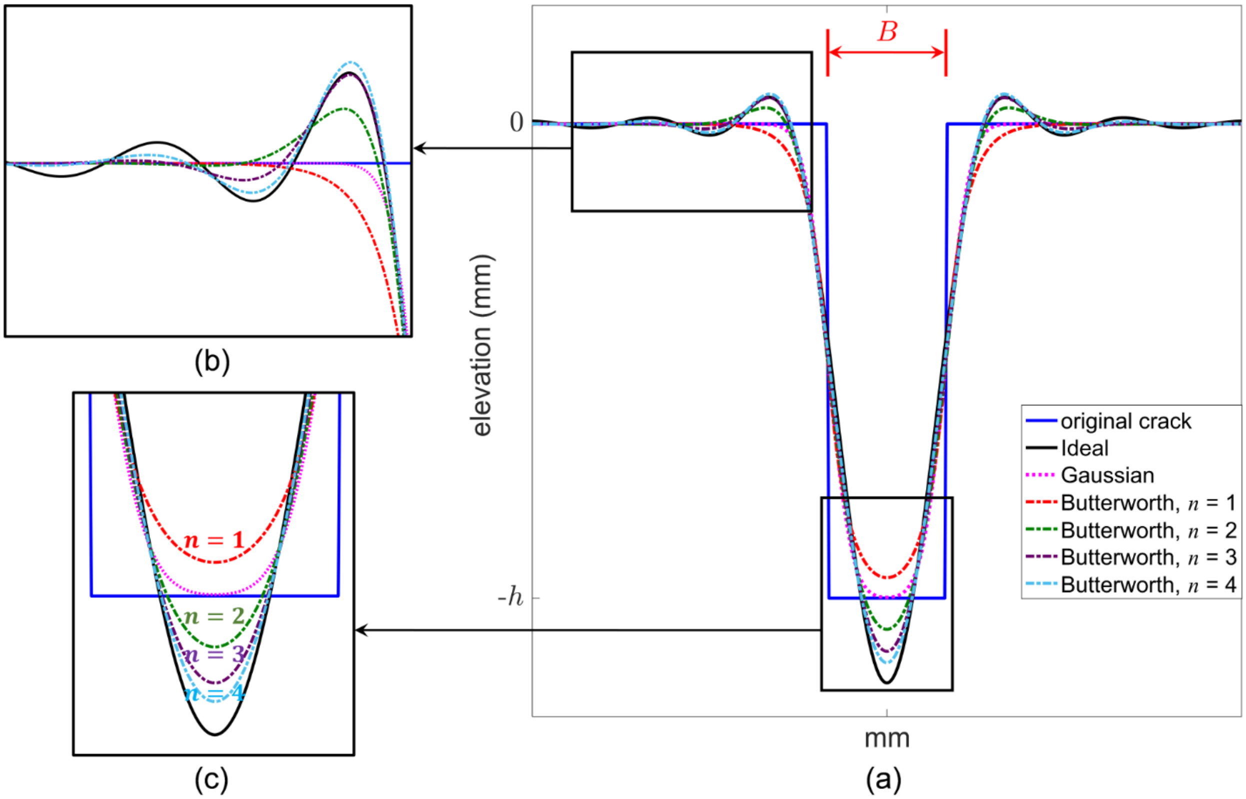


Figure 3-5. Zoomed-in views: (a) the filtered crack surface; (b) the edge of the filtered crack; and (c) the bottom of the filtered crack.

To demonstrate their performance on preserving the cracking information, the above low-pass filters are applied to the simulated cracked surface [Figure 3-4 (a)], and the filtering results are demonstrated in Figure 3-5. According to the derived relationship between the crack width and its frequency domain information, the cutoff frequency for the low-pass filters is selected as $\frac{1}{B}$ cycle/mm. In Figure 3-5 (a), the original cracked surface is compared with the surfaces filtered by the Ideal filter, the Gaussian filter, and the Butterworth filters with different orders. It can be observed from Figure 3-5 (a), that all three types of filters are capable of preserving the crack by assigning the cutoff frequency based on the derived relationship. However, as can be observed from the zoomed-in view in Figure 3-5 (b), the Ideal filter incurs ringing artifact on the crack edge. In addition, ringing starts to appear on the surfaces filtered by the Butterworth filters with order ≥ 3, but it is less obvious than that incurred by the Ideal filter. Meanwhile, no ringing artifact is produced in the filtered surface using the Gaussian filter. From both the Figure 3-5 (b-c), as the order of the Butterworth filter increases, the shape of the filtered crack approaches to that by the Ideal filter. At lower order ($n = 1, 2$), the Butterworth filters do not cause ringing in the filtered surface, showing similar performance with the Gaussian filter. But, it is worth noting that, by applying the Ideal filter and the Butterworth filter with order ≥ 2, the filtered crack depth is undesirably exaggerated [Figure 3-5 (c)].

In short, all three types of filters are capable of preserving the crack by properly assigning the cutoff frequency according to the derived relationship; among these filters, the Gaussian filter neither produces ringing artifacts in the filtered surface, nor exaggerates the depth of the filtered crack, unlike what the other two types of filters do. Although the above findings are obtained using the low-pass filter as a demonstration example, the ringing behavior in the transition band and the magnitude mismatch observed for the ideal filter and the Butterworth filter are also applicable in the high-pass and the notch filter cases [117]. Based on the above findings, the Gaussian filter is utilized in the filtering process in this study.

- High-pass filtering to eliminate surface variations

Road surface usually contains variations caused by rutting, uneven lanes, and vertical vibration of the data acquisition vehicle, as illustrated in Figure 3-1. The existence of these surface variations might deteriorate the crack detection performance. Conventional methods such as median filtering [52,53,114] and surface fitting [36] are adopted for background correction and crack enhancement purposes, but they often require subjective parameters (e.g., kernel size of the spatial domain filters). The frequency domain filtering applied in this methodology, however, follows the physical relationship between the frequency domain information and crack width to determine the filter parameters (e.g., cutoff frequency). Therefore, the proposed frequency domain filtering can provide robust and consistent results in practice and is independent from images or operating personnel.

In this study, the Gaussian high-pass image filter is selected to eliminate the low frequency surface variations, because it does not incur ringing artifacts in the filtered surface, due to its identical non-oscillatory form in both the frequency and spatial domain. The Gaussian high-pass image filter is expressed as Equation (3-5). The formulation is modified from that in [116] to account for the possible different pixel resolutions during 2D filtering of image data.

$$H_h(u,v) = 1 - e^{-\frac{\frac{u^2}{p_u^2}+\frac{v^2}{p_v^2}}{2D_h^2}} \tag{3-5}$$

where H_h denotes the Gaussian high-pass filter; u, v are the spectral coordinates along longitudinal and transverse direction, respectively, ranging from $-\frac{1}{2}$ to $\frac{1}{2}$ cycle/pixel; p_u and p_v are the pixel resolutions (unit: mm/pixel) along longitudinal and transverse direction, respectively; $D_h = \frac{1}{B_{max}}$ (unit: cycle/mm) is the high-pass cutoff frequency, where B_{max} (unit: mm) is the maximum width of the cracks to be detected.

- Low-pass filtering to suppress image noises

Image noises can occur during image acquisition, coding, transmission, and processing stage, and are usually shown as random variations of brightness or color [118]. On roadway surface, noises created by varying illumination condition, shadows, blemishes, concrete spall, etc., often bring difficulties in image-based crack detection [119]. Therefore, one critical step in image pre-processing is to suppress the image noises for improved crack detection performance. Noise is usually manifested as the high frequency content in an image, because it involves sharp changes in brightness or color. Based on the findings in section 3.1.4, the Gaussian low-pass image filter, expressed as Equation (3-6) [116], and modified similarly as Equation (3-5), is applied to suppress the high frequency image noises in this study.

$$H_l(u,v) = e^{-\frac{\frac{u^2}{p_u^2}+\frac{v^2}{p_v^2}}{2D_l^2}} \tag{3-6}$$

in the above expression, H_l denotes the Gaussian low-pass filter; $D_l = \frac{1}{B_{min}}$ (unit: cycle/mm) is the low-pass cutoff frequency, where B_{min} (unit: mm) is the minimum width of the cracks to be detected.

- Multiple notch filtering to remove grooves

Groove is a non-crack periodic pattern of a uniform depth, width, and shape on roadway surface. Surface grooving is an effective means to increase road traction as well as prevent vehicle sideways skidding, reduce hydroplaning, and provide effective braking surface [120-122]. However, in the presence of grooves, detection of cracks is particularly difficult, as the cracks have similar appearance to the grooves [123]. For example, grooves are often misidentified as cracks in the process of crack detection, because the groove width and depth are similar to those of cracks [124]. Figure 3-6 (a) illustrates a longitudinal grooved pattern in a cracked surface. In frequency domain, this periodic pattern shows as frequency harmonics [Figure 3-6 (b)], which can be effectively removed by applying notch filtering. A Gaussian multiple notch filter is designed in this methodology to remove grooved patterns in the image surface. The choice to adopt the Gaussian filter for notch filtering is for the same reason described in section 3.1.4. By implementing the multiple notch filter (Figure 3-7), the grooves can be effectively eliminated without losing cracking information. The Gaussian multiple notch filter proposed in this study is expressed as Equation (3-7).

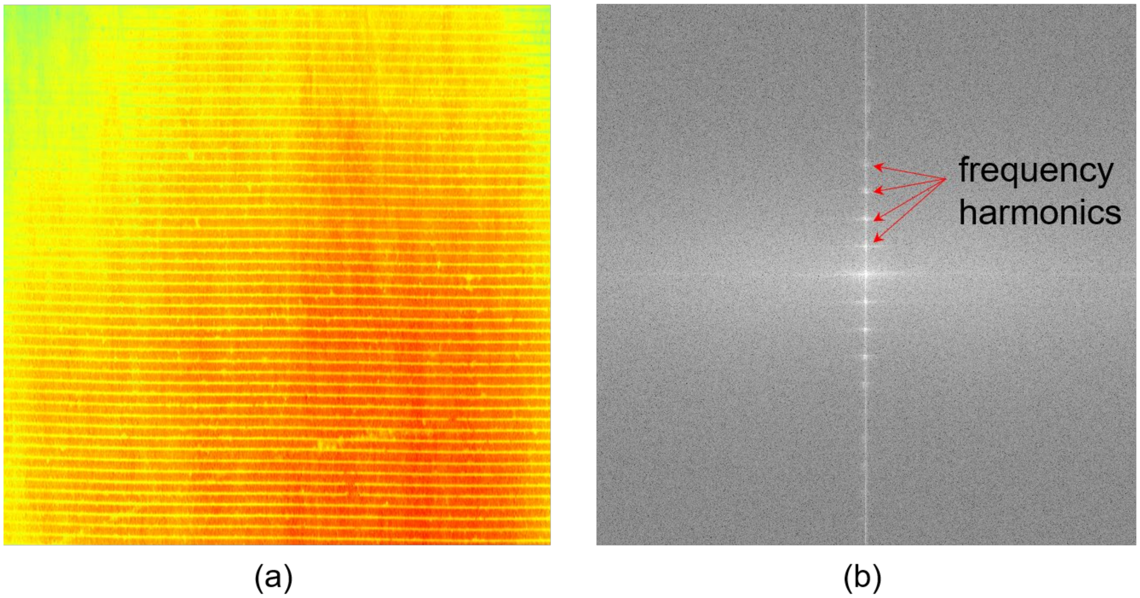


Figure 3-6. Grooves in a cracked surface: (a) spatial domain; and (b) frequency domain.

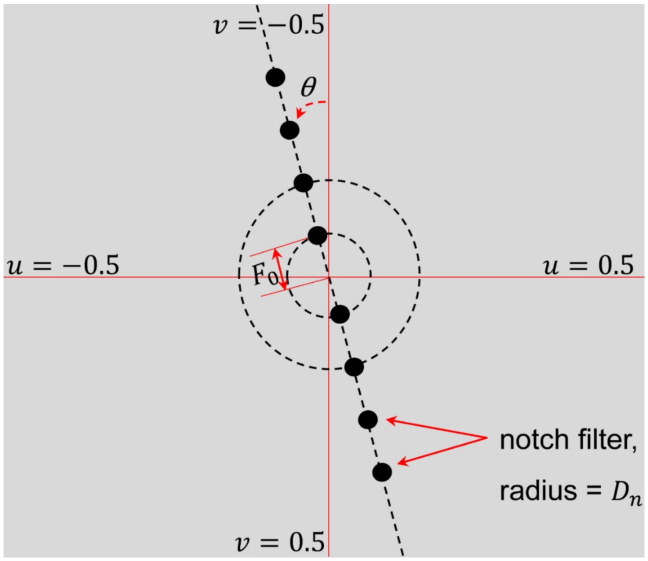


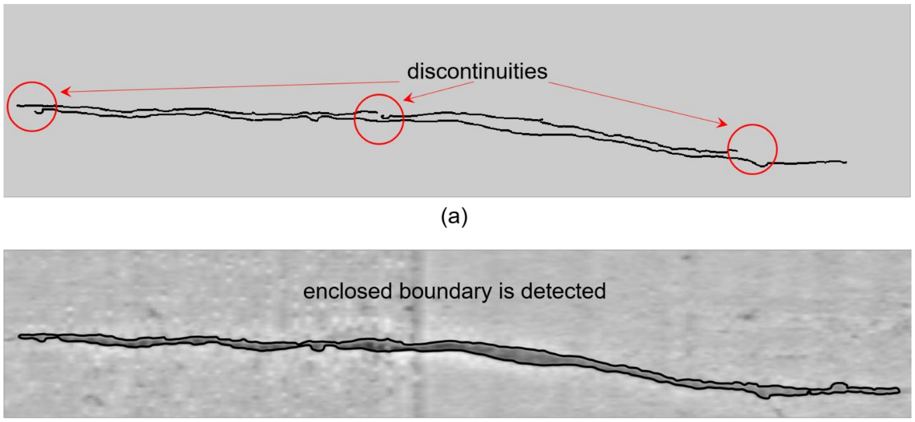
Figure 3-7. Multiple notch filter.

$$H_n(u,v) = \prod_{q=-Q,\, q\neq 0}^{Q} \left[1 - e^{-\frac{\frac{(u-qF_0 sin\theta)^2}{p_u^2}+\frac{(v-qF_0 cos\theta)^2}{p_v^2}}{2D_n^2}} \right] \qquad \textbf{(3-7)}$$

where H_n denotes the Gaussian multiple notch filter; q is a non-zero integer ranging from $-Q$ to Q, where Q refers to the number of frequency harmonics to be removed; $F_0 = \frac{1}{g}$ is the fundamental frequency, where g (unit: mm) denotes the groove spacing; D_n denotes the frequency radius of each circle-shaped notch filter (Figure 3-7); θ denotes the orientation of the grooved pattern, measured counterclockwise with respect to the longitudinal direction; for longitudinal groove, $\theta = 0^{\circ}$; for transverse groove, $\theta = 90^{\circ}$.

3.1.5 *Crack Detection Based on Contouring Analysis*

Conventional edge detection techniques have the drawbacks as being very sensitive to image noises [107]. Moreover, they do not consider crack connectivity and boundary [37], bringing difficulties to extracting each individual crack. The contouring analysis adopted in this methodology, however, can detect an enclosed crack boundary, lending itself to accurate estimation on cracking properties such as area, perimeter, etc. In an illustrative example shown as Figure 3-8, a crack is detected by both Canny edge detection [Figure 3-8 (a)] and the contouring analysis [Figure 3-8 (b)]. It is worth noting that in Figure 3-8 (b), the detected contour is superimposed on the cracked range surface. While discontinuities [circled in red in Figure 3-8 (a)] in crack boundary are observed in Canny edge detection result, an enclosed crack boundary [shown as a black contour in Figure 3-8 (b)] is accurately extracted by using the contouring analysis. The assumption of using the contouring analysis for surface crack detection is that, surface cracks have different elevations than their surrounding areas. This assumption, however, has a limitation that the performance might be deteriorated when detecting shallow cracks where the reduction of elevation is not significant enough.



(a)

(b)

Figure 3-8. A crack detection example using Canny edge detection and the contouring analysis: (a) detected crack by Canny edge detection; and (b) detected crack by the contouring analysis.

- Marching squares algorithm for contour detection

In the proposed methodology, a popular and extensively used computer algorithm for boundary detection (i.e., marching squares [125-127]) is employed to extract crack contour candidates from the image surface. The marching squares algorithm gives a piece-wise approximation to the boundary of a 2D object. First, the image is represented by a grid of 2-by-2 pixel cells. This method analyzes the local properties of each cell, which has a finite set of cell configurations [126]. Then, the object can be enclosed by these cells through linear interpolation over the cell interval.

The parameters in the marching squares algorithm are the contour levels. In this methodology, a single contour level is set based on the crack depth to be detected. In this study, the crack depth is determined as $h = -1\ mm$. In civil engineering practice, the crack depth to be detected can be determined by road survey companies or state transportation agencies.

It is worth noting that applying such a boundary detection algorithm alone will yield erroneous crack detection result, because: *i*) without a proper filtering process to remove the surface variations, the determination of the single contour level is dependent on the elevation of the image surface rather than the crack depth to be detected; and *ii*) additional logics and criteria are required to distinguish the crack contours from non-crack contours such as noise-induced small contours and pothole contours.

- Contour qualification and crack classification

As illustrated in Figure 3-2, after the contours are extracted from the filtered image surface using the marching squares algorithm, a set of logics and criteria is developed in this methodology for crack detection and classification purpose. The detailed four-step procedure is explained as below:

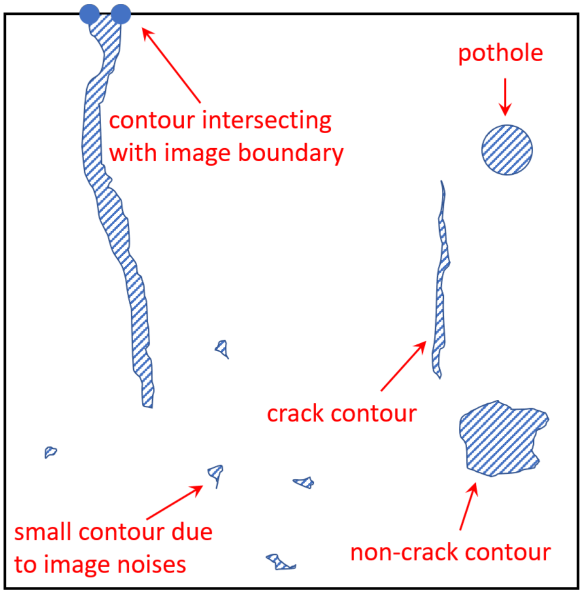


Figure 3-9. Different contour scenarios (the shaded areas indicate lower elevation than the contour level).

i) Enclose the contours intersecting with image boundary

The contours that have intersections with the image boundary will not be enclosed, as shown in Figure 3-9. Therefore, a criterion is developed to address this issue: if the starting and end contour points of a contour lie in the image boundary, then these two points are connected to produce an enclosed boundary.

ii) Extract the contour properties

The contour properties including area, perimeter, and centroid are calculated for further analysis.

iii) Remove the contours with small area

Despite the low-pass filtering process, a few remnant image noises still exist in the filtered range image. Such image noises might be detected as small contours, as shown in Figure 3-9. By using

Equation (3-8), the small contours due to image noises can be removed. In this equation, A_i is the area of the i^{th} contour candidate, and tol_1 is the tolerance value for the smallest contour area. The value of $tol_1$is a tradeoff between computational efficiency and detection performance. In this study, the choice of $tol_1 = 100$ mm^2, determined through experiments on a large set of range image data, successfully ensures the removal of the majority of noise-induced small contours, as can be illustrated through the subsequent experimental study. In practice, the tolerance value can be determined by road survey companies or state transportation agencies.

Criterion 1: small contours are removed upon meeting this criterion

$$A_i < tol_1 \tag{3-8}$$

iv) Crack classification

The crack classification is performed to distinguish the real cracks from those non-crack contours (Figure 3-9). Surface cracks usually have slender shapes; therefore, it is reasonable to distinguish the crack and non-crack contours based on this attribute. Equation (3-9) is proposed to disqualify the contours with non-slender shapes. In this equation, p_i denotes the perimeter of the i^{th} contour.

Criterion 2: Non-slender contours are removed upon meeting this criterion

$$\frac{4A_i}{p_i^2} > tol_2 \tag{3-9}$$

Generally, the value of $\frac{4A_i}{p_i^2}$ in Equation (3-9) reduces as a contour becomes slenderer in shape. For example, if a rectangular-like contour has little width with respect to its height, then the value of $\frac{4A_i}{p_i^2}$ becomes very small, and Equation (3-9) will not hold. As a result, the corresponding contour will be identified as a real crack. For a square-shaped contour, $\frac{4A_i}{p_i^2}$ =0.25; for a round-shaped contour which resembles a pothole (Figure 3-9), this value is equal to $\frac{1}{\pi}\approx$0.3. In this study, the threshold value $tol_2 = 0.2$, determined through experiments on a large set of range image data; and, this choice has enabled successful crack detection in the subsequent experimental study. In practice, this threshold value can also be determined by road survey companies or state transportation agencies.

3.2 Deep Learning-Based Crack Classification

3.2.1 *Motivation*

Despite many successful developments of DCNN-based methods on crack detection as reviewed in section 2.2.2, some issues have yet to be properly addressed. One issue is related with the image data. Albeit the high adaptability of DCNN, directly applying this type of methods to realistic situations can still be very difficult, due to the high irregularity and complexity in the image data.

Therefore, it is common to employ image pre-processing to reduce the image complexity by noise removal and crack enhancement, as adopted by many DCNN-based methods [30,41,42]. However, it is more practical and preferable to directly learn from the raw data (possibly noise-contaminated) upon applying DCNN for crack detection to avoid additional human intervention. Moreover, typical non-crack patterns including grooves, pavement edges, and shoulder drop-offs can be misidentified as cracks due to their similarities, resulting in deterioration of the crack detection performance, as reported by [30]. Thus, it remains a challenge even for DCNN-based crack detection methods to reach consistent performance when directly using raw data with possible image contaminations.

Another issue lies in the DCNN architecture. It is critical to determine the design configurations and training strategies for a crack classification DCNN architecture. For example, selection of the hyperparameters related with network structure (e.g., kernel size, network width and depth) and training (e.g., mini-batch size and learning rate) can impact the accuracy and efficiency of the DCNN architecture up to a significant extent. Faghih-Roohi, et al. [128] implemented three DCNN architectures (small, medium, and large in terms of network width and depth) with various combinations of hyperparameters for detection of defects on rail surfaces. They observed that the large model outperformed the rest by a margin of 0.5% on both the accuracy and F1 score [104], but it required a longer computational time. Pauly, et al. [129] evaluated the performance metrics by varying the network depth, and they showed that the network with a deeper architecture led to up to a 2% improvement on the detection accuracy. Tong, et al. [30] compared the classification performance on range image data between the CrackNet II and its three variants differing in the size of the kernels. The CrackNet II with smallest kernel sizes had a 20% higher F1 score than the one with the largest kernel sizes. They concluded that increasing kernel size resulted in worse performance.

As to the selection of the hyperparameters for training, it has been reported that the process of configuring and choosing adequate hyperparameters is quite tedious, and no exact guidelines for those hyperparameter optimizations are available [39]. The design process of an ideal DCNN architecture for a task (i.e., crack classification in this context) must be conducted via experimentation guided by monitoring the validation set error [38]. Nevertheless, partially due to the lack of publicly available laser-scanned roadway range image datasets with high diversities (e.g., grooved patterns, various crack patterns), there is very few study discussing the impact from hyperparameter selection upon applying DCNN-based crack classification on this specific type of data.

To address the above issues and promote a practical and robust solution for deep learning-based roadway crack classification on laser-scanned range images, the proposed methodology investigates the influence from hyperparameter selection through experiments on range image data of high diversity and irregularity. A series of DCNN classifiers with varying hyperparameter configurations in terms of their architecture layouts and training settings are developed and evaluated through quantitative measures. Among a family of 36 proposed DCNN classifiers, the optimal architecture that can best describe and reflect the complexity of the laser-scanned roadway images by achieving the highest classification performance is thus determined. The proposed classifier does not require any pre-processing on the raw range images, therefore it is robust against image noises and non-crack patterns such as grooves, pavement edges, and shoulder drop-offs. The contribution of the proposed methodology is three-folded:

i) A hyperparameter selection process is developed for DCNN-based roadway crack classification using laser-scanned range images. Consistent observations regarding the impact from hyperparameter selection are observed upon evaluating the DCNN performance on range image data of high complexities. And, the conclusions can provide prior knowledge for DCNN designs in similar applications using laser-scanned roadway range images;

ii) The optimal architecture with associated training configuration determined through the hyperparameter selection process can achieve accurate and robust classification performance on different range image datasets with diverse patterns and image disturbances;

iii) A laser-scanned roadway range image dataset (LRRD) [130] on asphalt and concrete pavements is collected and made publicly available to benefit the community.

Thus, through this methodology, Challenge 5 (section 1.2) regarding the issue of hyperparameter selection can be properly tackled. Besides, as this proposed methodology directly exploits raw range image rather than filtered image data for analysis, it also addresses Challenges 2 and 3 by introducing robustness to the crack detection.

3.2.2 *Proposed Methodology*

- Proposed DCNN architecture

One of the objectives in this study is to explore the impacts from varying hyperparameters on the network performance, and then determine the optimal hyperparameter configuration through a series of experiments. Thus, multiple DCNN architectures with different layouts/specifications are investigated. For example, the candidate shown in Figure 3-10 is a 7-layer architecture (No. 25 in Table 4-3) which is analyzed in the experimental cases. In this figure, L1 through L6 are convolutional blocks, and L7 is a fully connected layer; "BN" and "LReLU" in the convolutional blocks represent batch normalization layer and leaky rectified linear unit (LReLU) layer, respectively; "Dropout" refers to a dropout layer to prevent overfitting and improve generalization; "Softmax" corresponds to a softmax normalization layer. In this example, the convolutional blocks L1 through L5 use convolutional kernels with a stride of 2 for feature extraction; in L6, 1×1 convolutional kernels with a stride of 1 are used for cross channel pooling. Zero paddings are assigned to produce output feature maps with a desired dimension. The definitions of these network layers can be found in section 2.3.1. The proposed DCNN classifier predicts the input raw range images as whether containing cracks or not. Detailed specifications on the network architecture are tabulated in Table 4-3.

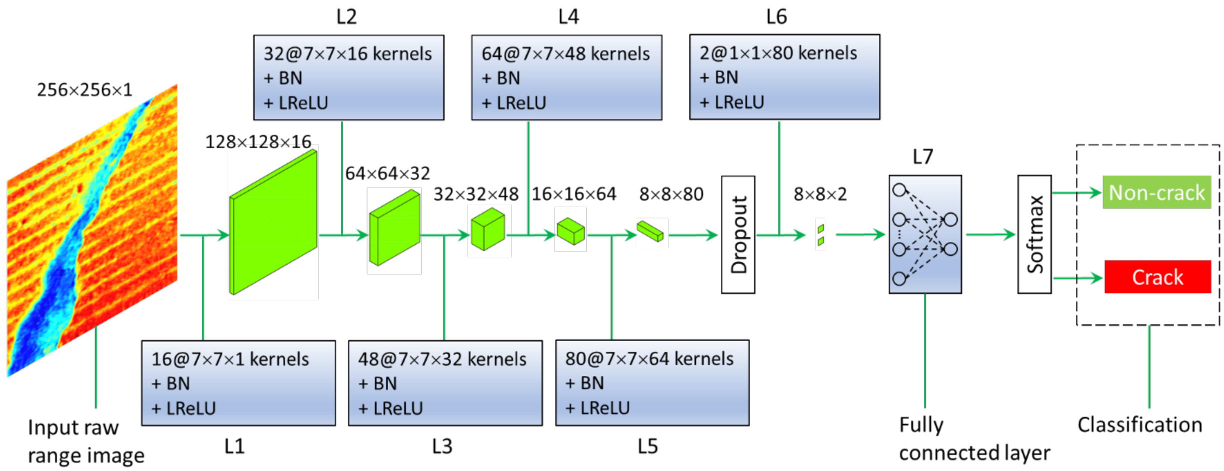


Figure 3-10. A DCNN architecture candidate for performance evaluation.

- Hyperparameter selection

Hyperparameters in a neural network are the variables which are determined prior to training; in contrast, the parameters including weights and biases in a network are updated during training. The hyperparameters can be mainly categorized into two types: those related with network structure, and those related with training. This section describes the hyperparameters investigated in this study.

i) Hyperparameters related with network structure

Hyperparameters such as kernel size, stride, network width (i.e., number of kernels in each convolutional layer), and network depth (i.e., number of layers) determine the layout of a convolutional neural network. As explained in section 3.2.1, different configurations on these hyperparameters can vary the network performance to a significant level. For example, by increasing the network depth, a neural network can learn more complex features, but meanwhile it might be more compute-intensive and can potentially suffer from overfitting. Therefore, these hyperparameters need to be properly selected through experiments. In this methodology, an experimental study is designed to explore the impacts from three hyperparameters, which are kernel size, network width, and network depth. It has been reported in the literature [30,128,129] that these three hyperparameters have a notable influence on the network performance. Case I of the experimental study, which will be introduced later, compares the performance of different architectures with varying kernel sizes, network widths, and network depths. Based on the comparison, the optimal hyperparameter configuration that yields the most efficient and accurate classification performance is determined.

ii) Hyperparameters related with training

In this study, the mini-batch SGD with momentum algorithm, as described in section 2.3.4, is adopted as the optimization technique. Thus, several hyperparameters including weight decay factor, momentum, number of epochs, mini-batch size, learning rate and learning rate drop factor, dropout factor, and LReLU factor are involved in the training process. While no exact guidelines are available for configuring and choosing these hyperparameters [39] for crack classification tasks, Case II of this experimental study focuses on investigating the optimal joint specification of the following hyperparameters: mini-batch size, learning rate, dropout factor, and LReLU factor. In Case II, to confine the discussion to these hyperparameters, the values of the other hyperparameters such as weight decay are fixed to be the same as used in other successful applications [30,39].

- Flow chart of the proposed DCNN-based crack classification methodology

The flow chart of the proposed DCNN-based crack classification methodology which contains a two-step procedure (i.e., training and prediction) is illustrated in Figure 3-11. First, the proposed DCNN classifier is trained and validated on range image data; then, the trained classifier is utilized for crack classification on new datasets. By using the sliding window technique as introduced in section 2.3.2, the collected pavement image is cropped into many patches with smaller sizes, which reduces computational cost during training. Meanwhile, each cropped image patch carries not only the image feature but also location information. Once the image patches are predicted by the DCNN classifier, an inverse operation of the sliding window technique restores each patch to its original location, indicating the corresponding region as containing cracks or not. Thus, a crack map containing crack location information can be generated.

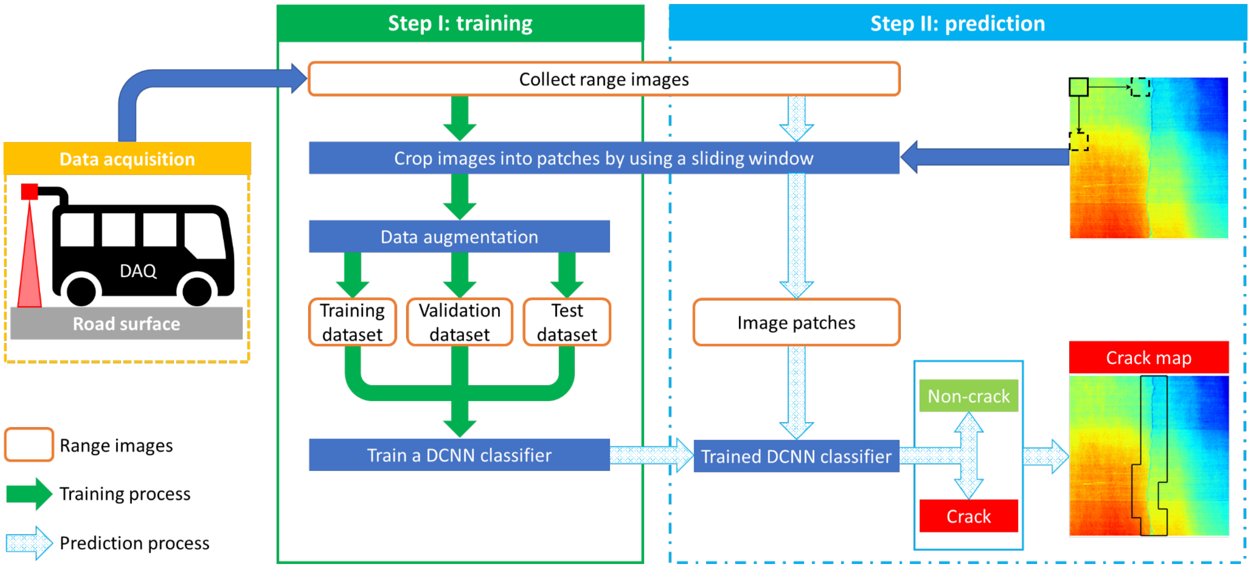


Figure 3-11. Flow chart of the proposed crack detection methodology based on range images and DCNN.

3.3 Deep Learning-Based Crack Segmentation

3.3.1 *Motivation*

As reported in section 2.2.2, a majority of current DCNN-based crack segmentation applications only used intensity images for training, thus their performance may potentially deteriorate due to disturbances in intensity image data such as shadows, uneven lighting condition [19], oil spills and blemishes [29,131].

In the recent decade, researchers explored laser-scanned range image data for crack detection, relying on the elevation difference between cracks and non-crack regions to interpret the presence of cracks [19]. Zhang, et al. [42], Tong, et al. [30], and Fei, et al. [40] developed CrackNet and its variants, which are DCNNs trained and tested on laser-scanned range images, for roadway crack segmentation. However, while this type of image data has promising advantages over traditional intensity images such as being insensitive to changing illumination, it also suffers from issues including surface variations and non-crack patterns (e.g., grooves), which need to be properly addressed prior to crack detection [7,29]. In fact, due to these real-world complexities in range image data, most of the current DCNN-based crack segmentation methods [30,40,42] still employ image pre-processing (e.g., surface flattening, median filtering) on range images for crack enhancement. Besides, as stated in [30], issues due to non-crack patterns such as grooved patterns can deteriorate the segmentation performance, where the CrackNet II misidentified some grooves in concrete surfaces as cracks. The grooved pattern is a man-made pattern with a uniform depth, width, and shape on roadway surfaces to increase traction and resist sideways skidding. Because pavement grooves often possess similar features as cracks, such as width and depth, they can bring additional disturbances and uncertainties to image-based crack detection [29]. For current DCNN-based crack segmentation methods using roadway range images, it remains a challenge to achieve consistent and robust segmentation performance under the contamination of grooves.

A novel methodology is proposed based on encoder-decoder DCNNs for pixel-wise crack classification on roadway range images, under possible contamination of disturbances, such as surface variations and grooved patterns. Its contributions can be summarized in the following aspects:

i) The proposed DCNN methodology utilizes range images for roadway crack segmentation, and it does not require any image pre-processing techniques to address the image issues including surface variations and pavement grooves;

ii) A series of encoder-decoder networks are proposed and evaluated through a comparative study to investigate the impacts on crack segmentation performance; through the comparative study, the influence of residual connections on DCNN-based crack segmentation using roadway range images is demonstrated;

iii) The optimal architecture determined in this study can achieve robust and consistent segmentation performance on laser-scanned roadway range images contaminated by image disturbances such as surface variations and grooved patterns.

By achieving the above contributions by this DCNN-based crack segmentation methodology, the issues as described in Challenge 6 in section 1.2 can be properly addressed.

3.3.2 *Proposed Methodology*

A set of encoder-decoder networks with varying network depths and residual connections are proposed in this study for comparison. It is noted that, the impacts of residual connections (section 2.3.5) on crack segmentation performance of DCNNs using laser-scanned roadway images, which has not been thoroughly investigated in literature, are demonstrated and evaluated in the experimental section.

Six encoder-decoder networks, denoted as Net-1 through 6, with gradually increased network depths, are proposed for crack segmentation. Meanwhile, the number of residual connections in these proposed encoder-decoder networks increases from 1 to 6, correspondingly.

To isolate the influences from increasing network depth and from adding residual connections, an additional series of six "plain" counterparts (i.e., without residual connections) of Net-1 through 6 are constructed for comparison.

- Encoder-decoder networks with residual connections

Six encoder-decoder networks with residual connections, denoted as Net-1 through 6, are proposed in this study. In Figure 3-12 which displays their basic layout, the red dashed box represents architecture-specific layers which are further illustrated in Figure 3-13 (a-f), respectively, for Net-1 through 6. As shown in these two figures, the encoder consists of multiple convolutional blocks for feature extraction. Each convolutional block consists of a 2×2 max pooling layer with a stride of 2 for down-sampling, a 3×3 convolutional layer which adopts 3×3 kernels with a stride of 1 and padding of 1 for feature extraction, a batch normalization layer to improve model generalization, and a LReLU layer to add nonlinearity. For each transposed convolutional block in the decoder, a 3×3 transposed convolutional layer utilizing 3×3 kernels with a stride of 2 and cropping of 1 is adopted for up-sampling, followed by a batch normalization and LReLU layer. The 1×1 convolutional layer performs cross-channel pooling to produce an output map of a desired depth. The softmax layer placed at the end produces a probability map for each input image. The output dimension is shown along with each layer. In each architecture displayed in Figure 3-12 combined with Figure 3-13, low-level features extracted from the encoder are added to high-level features generated by the decoder through residual connections. These six architectures (Net-1 through 6) have different model complexities, represented by the increased network depths and associated number of parameters as shown in Table 3-1.

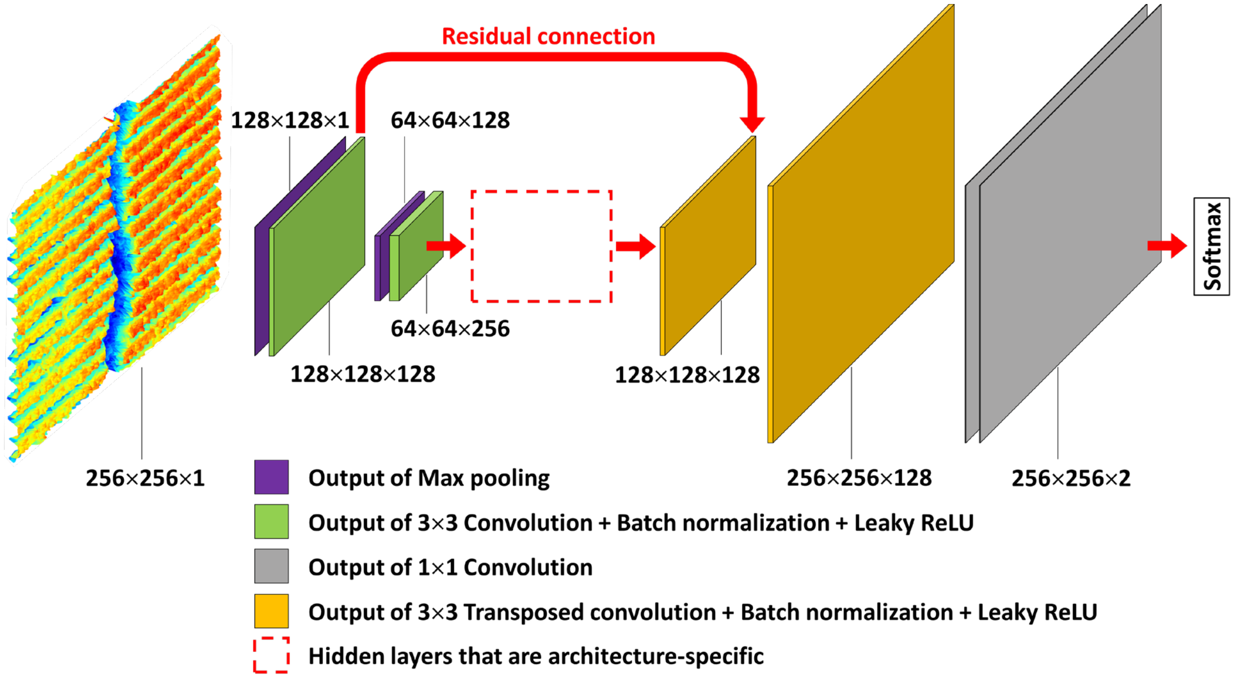

Residual connection
128×128×1
64×64×128
64×64×256
128×128×128
128×128×128
256×256×1
256×256×128
256×256×2
Softmax
Output of Max pooling
Output of 3×3 Convolution + Batch normalization + Leaky ReLU
Output of 1×1 Convolution
Output of 3×3 Transposed convolution + Batch normalization + Leaky ReLU
Hidden layers that are architecture-specific

Figure 3-12. Overall layout of the proposed DCNN architectures for semantic segmentation.

Table 3-1. Detailed configurations of the proposed architectures.

Index	Net name	Network depth*	Residual connection	Number of residual connections	Number of parameters ($\times 10^6$)
1	Net-1	6	√	1	0.45
2	Net-2	8	√	2	1.93
3	Net-3	10	√	3	7.84
4	Net-4	12	√	4	31.45
5	Net-5	14	√	5	88.10
6	Net-6	16	√	6	163.61
7	Net-1A	6	×	N/A	0.45
8	Net-2A	8	×	N/A	1.93
9	Net-3A	10	×	N/A	7.84
10	Net-4A	12	×	N/A	31.45
11	Net-5A	14	×	N/A	88.10
12	Net-6A	16	×	N/A	163.61

*: total number of the convolutional and transposed convolutional layers.

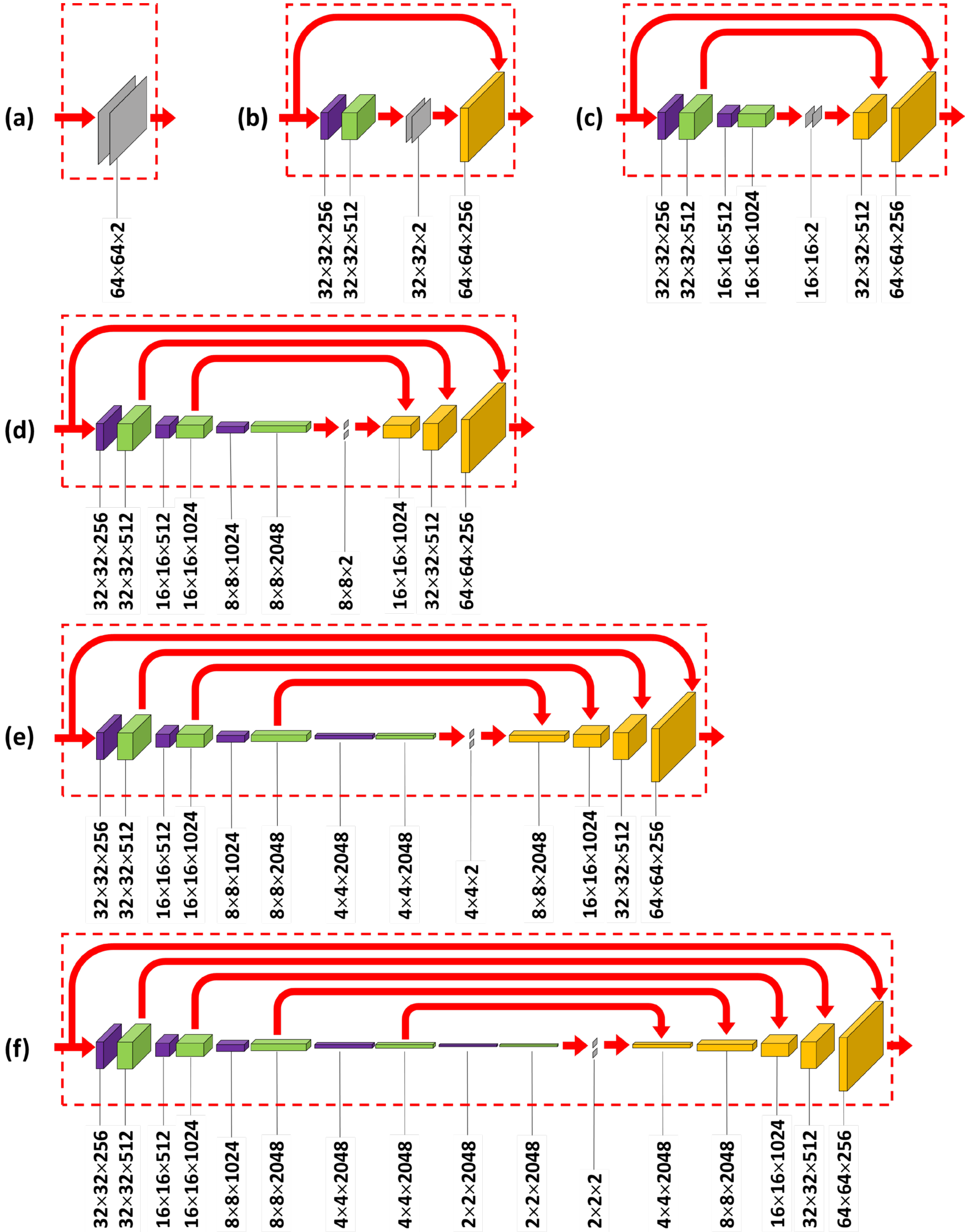


Figure 3-13. Hidden layers of the proposed architectures: (a-f) Net-1 through 6.

- Encoder-decoder networks without residual connections

In addition to Net-1 through 6, another six architectures denoted as Net-1A through 6A are constructed as their “plain” counterparts (see Table 3-1), which have the same layer configurations except they do not contain any residual connections. Thus, through a comparison between each network and its “plain” counterpart, the effect from residual connections on crack segmentation performance can be demonstrated. Figure 3-14 shows the architecture of Net-4A as an illustrative example. The detailed layer configuration for Net-4A is tabulated in Table 3-2. For concision, similar configuration information for the other architectures is not presented in this study.

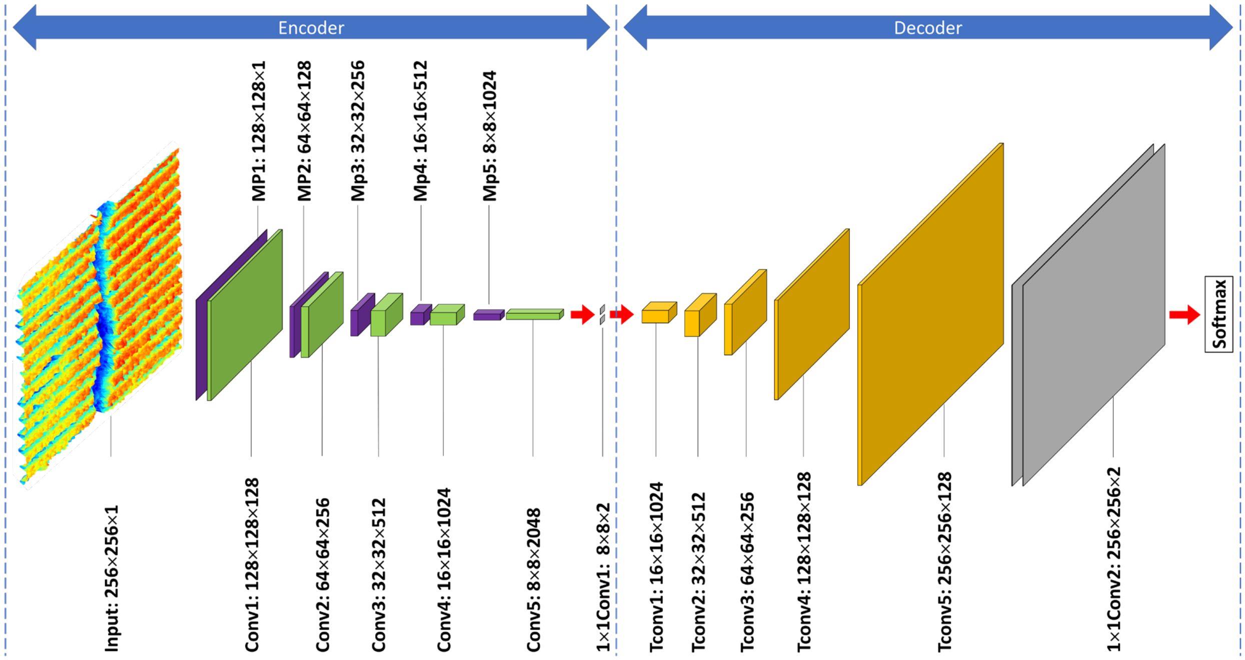


Figure 3-14. Net-4A: a counterpart of Net-4 (without residual connections).

Table 3-2. Detailed layer configuration of Net-4A.

Layer name	Layer type**	Output dimension	Kernel size	Depth	Stride	Padding*	Cropping*	Learnable parameters			
Input	INPUT	256×256×1	-	-	-	-	-	-			
MP1	MaxPool	128×128×1	2×2	-	2	0	-	-			
Conv1	CONV	128×128×128	3×3×1	128	1	1	-	weight	3×3×1×128	bias	1×128
BN1	BN	128×128×128	-	-	-	-	-	scale	1×128	shift	1×128
LReLU1	LReLU	128×128×128	-	-	-	-	-	-			
MP2	MaxPool	64×64×128	2×2	-	2	0	-	-			
Conv2	CONV	64×64×256	3×3×128	256	1	1	-	weight	3×3×128×256	bias	1×256
BN2	BN	64×64×256	-	-	-	-	-	scale	1×256	shift	1×256
LReLU2	LReLU	64×64×256	-	-	-	-	-	-			
MP3	MaxPool	32×32×256	2×2	-	2	0	-	-			
Conv3	CONV	32×32×512	3×3×256	512	1	1	-	weight	3×3×256×512	bias	1×512
BN3	BN	32×32×512	-	-	-	-	-	scale	1×512	shift	1×512
LReLU3	LReLU	32×32×512	-	-	-	-	-	-			
MP4	MaxPool	16×16×512	2×2	-	2	0	-	-			
Conv4	CONV	16×16×1024	3×3×512	1024	1	1	-	weight	3×3×512×1024	bias	1×1024
BN4	BN	16×16×1024	-	-	-	-	-	scale	1×1024	shift	1×1024
LReLU4	LReLU	16×16×1024	-	-	-	-	-	-			
MP5	MaxPool	8×8×1024	2×2	-	2	0	-	-			
Conv5	CONV	8×8×2048	3×3×1024	2048	1	1	-	weight	3×3×1024×2048	bias	1×2048
BN5	BN	8×8×2048	-	-	-	-	-	scale	1×2048	shift	1×2048
LReLU5	LReLU	8×8×2048	-	-	-	-	-	-			
1×1Conv1	CONV	8×8×2	1×1×2048	2	1	0	-	weight	1×1×2048×2	bias	1×2
TConv1	TransCONV	16×16×1024	3×3×2	1024	2	-	1	weight	3×3×2×1024	bias	1×1024
BN6	BN	16×16×1024	-	-	-	-	-	scale	1×1024	shift	1×1024
LReLU6	LReLU	16×16×1024	-	-	-	-	-	-			
TConv2	TransCONV	32×32×512	3×3×1024	512	2	-	1	weight	3×3×1024×512	bias	1×512
BN7	BN	32×32×512	-	-	-	-	-	scale	1×512	shift	1×512
LReLU7	LReLU	32×32×512	-	-	-	-	-	-			
TConv3	TransCONV	64×64×256	3×3×512	256	2	-	1	weight	3×3×512×256	bias	1×256
BN8	BN	64×64×256	-	-	-	-	-	scale	1×256	shift	1×256
LReLU8	LReLU	64×64×256	-	-	-	-	-	-			
TConv4	TransCONV	128×128×128	3×3×256	128	2	-	1	weight	3×3×256×128	bias	1×128
BN9	BN	128×128×128	-	-	-	-	-	scale	1×128	shift	1×128
LReLU9	LReLU	128×128×128	-	-	-	-	-	-			
TConv5	TransCONV	256×256×128	3×3×128	128	2	-	1	weight	3×3×128×128	bias	1×128
BN10	BN	256×256×128	-	-	-	-	-	scale	1×128	shift	1×128
LReLU10	LReLU	256×256×128	-	-	-	-	-	-			
1×1Conv2	CONV	256×256×2	1×1×128	2	1	0	-	weight	1×1×128×2	bias	1×2
Softmax	SOFTMAX	256×256×2	-	-	-	-	-	-			
								Sum	31450116		

*: padding is performed on all boundaries; cropping is performed on bottom and right boundaries.

**: INPUT: image input layer; MaxPool: max pooling layer; CONV: convolutional layer; BN: batch normalization layer; LReLU: leaky rectified linear unit; TransCONV: transposed convolutional layer; SOFTMAX: softmax normalization layer.

- Flow chart of the proposed DCNN-based crack segmentation methodology

Figure 3-15 illustrates the flow chart of the DCNN-based crack segmentation methodology proposed in this study, which is comprised of two phases: *i*) DCNN training; and *ii*) DCNN prediction. In DCNN training, first, range images are acquired from roadways; then, image patches with a dimension of 256×256 pixels are produced through the sliding window technique (section 2.3.2); ground truth labels are generated for the image patches containing cracks; finally, the cracked image patches with their ground truth labels are augmented and then utilized for DCNN training. Once the DCNN is trained, it can be used in the second phase, which is DCNN prediction. During prediction, new image data is acquired from roadway surfaces, cropped into patches using the sliding window technique; then, crack maps with pixel-wise resolution are generated by the DCNN for the image patches; finally, through an inverse operation of the sliding window technique, a crack map for the roadway image is reconstructed.

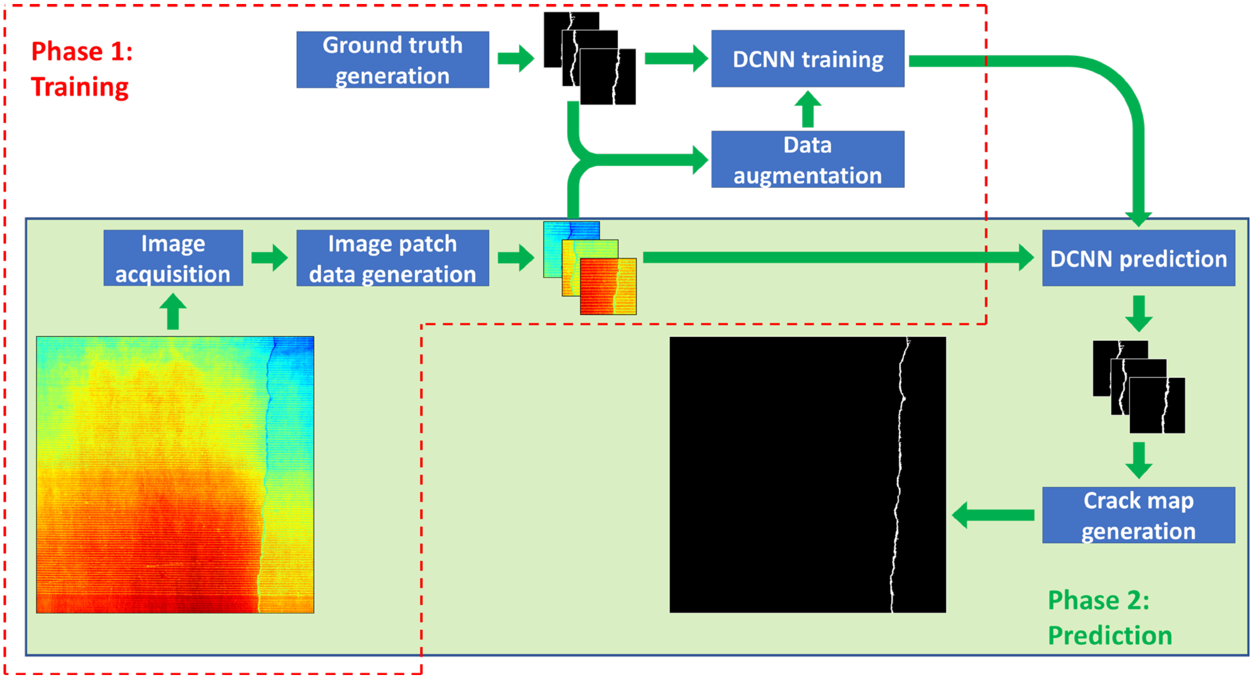


Figure 3-15. Flow chart of the proposed DCNN-based crack segmentation methodology.

3.4 Deep Learning-Based Data Fusion for Crack Detection

3.4.1 *Motivation*

As reviewed in section 2.2.2, current DCNN-based crack segmentation methods either used intensity or range image for analysis. Depending on the image type, crack detection methodologies may suffer from issues such as uneven illumination and low contrast in intensity images or surface variations and grooved patterns in range images. For example, as observed in [30], grooved patterns in concrete pavements resulted in false-positive detections. Besides, to resolve the disturbances existing in either type of image data, image pre-processing techniques such as background correction [41], line filtering [42], surface flattening [30], and median filtering [40] are adopted prior to DCNN training and testing. However, in practice, it is often unclear about the appropriate choices of the parameters associated with the pre-processing procedure; furthermore, certain level of expertise or prior knowledge is often required upon designing a pre-processing procedure, leading to a scenario where the effect of pre-processing may be user-dependent. Therefore, despite the wide applicability of DCNN-based methodologies, uncertainties or subjectivities due to parameter selection may still arise. For example, as reported in [40], it is difficult to determine the optimal kernel size of the median filter to eliminate the surface variations in the range image.

Instead of relying on image pre-processing techniques, which may bring uncertainties or subjectivities, the methodology proposed in this study explores the feasibility of fusing the raw intensity and range image to alleviate the image-related issues through cross-domain feature correlation. Data fusion is a terminology in informatics, referring to a process to obtain more comprehensive information and reduce uncertainty by integrating multiple data sources instead of considering each data source separately [43,132]. Such a fusion process can be applied to multiple data sources based upon a single type of data (i.e., homogeneous data fusion), or upon different types of data (i.e., heterogeneous data fusion) [43-45,133]. In the context of image-based crack detection, several applications using homogeneous data fusion such as [134,135] were reported in literature. However, the combined information from homogeneous data sources may still carry the issue existing in the same type of data [45]. For example, by fusing the intensity image data, which may suffer from uneven illumination, the fused image data may still reproduce the issue incurred by uneven illumination. Thus, it is preferable in practical situations to leverage heterogeneous data fusion to incorporate data of diverse characteristics and further reduce the uncertainty existing in each type of data [45,132,133]. In a study on near-surface crack detection, Heideklang and Shokouhi [136] applied heterogeneous data fusion to integrate information from three types of data to improve the detection performance. However, their method is not DCNN-based, and hence may have difficulties adapting to data with real-world complexities. Beckman, et al. [137] proposed a method using region-based DCNN and two types of data for concrete spalling detection. In their study, the DCNN architecture considers only one type of data sources—the intensity image data, while the other type of data (i.e., depth information through a depth sensor) is not incorporated into the architecture of the DCNN. Overall, in the existing literature on DCNN-based roadway crack segmentation, the feasibility and strategy of applying heterogeneous image fusion have yet to be investigated; besides, the effects from different types of image data on DCNN segmentation remain to be discovered and demonstrated.

In this study, the laser imaging system as introduced in section 2.4 is utilized to collect both intensity and range images with spatial correlation for analysis. Two novel DCNN-based methodologies with heterogeneous image data fusion for roadway crack classification (section 3.4.3) and segmentation (section 3.4.4), respectively, are proposed. The contributions are:

i) A novel heterogeneous image data fusion approach is proposed, by integrating the range and intensity image data based on the concept of hyperspectral imaging to offer robust crack detection performance;

ii) A series of new DCNNs, representing different strategies to exploit fused raw image data, are developed and compared for crack classification and segmentation tasks, respectively;

iii) To evaluate the impact on crack detection from heterogeneous image data, four types of data including raw range, raw intensity, filtered range (obtained through image pre-processing), and fused raw image are trained and tested on baseline DCNN architectures for comparison; additionally, based on the experimental analysis, findings related to the use of image pre-processing are provided.

Through the proposed methodologies with the associated image fusion strategies for crack classification and segmentation tasks, Challenge 7 (section 1.2) which is related with heterogeneous image data is addressed.

3.4.2 *Heterogeneous Image Data*

Four different types of image data are considered in this study, which are raw intensity, raw range, filtered range, and fused raw image. The characteristics of each type of image data are introduced in this section.

- Raw intensity image

Intensity image is used by many image-based techniques [6] for crack detection. The general assumption upon using the intensity image is that cracked regions have lower intensity values (i.e., darker) than non-crack regions. Thus, under the situation of changing illumination condition or low intensity contrast between crack and non-crack regions, the performance of intensity-based crack detection may deteriorate [19]. Moreover, image disturbances such as shadows, blemishes, and oil stains which also have low intensity values may add difficulty and uncertainty to crack detection on intensity images [19,131]. As an example, the intensity and range data of ten image samples are illustrated in Figure 3-16 to demonstrate the issue of low contrast in intensity images. In Figure 3-16, surface cracks can be clearly observed in the range image data; however, due to low intensity contrast between the cracks and non-crack regions, the cracks are not noticeable in the corresponding intensity images. Based on the cracks with low intensity contrast identified through this study, their intensity contrast is usually lower than 30 (pixel intensity ranges from 0~255).

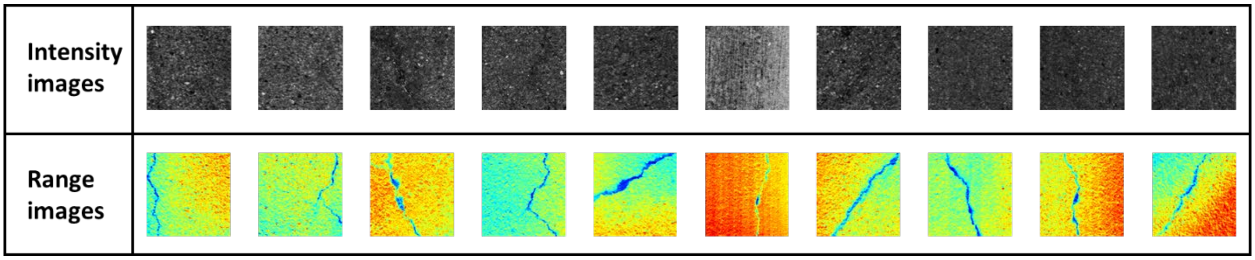


Figure 3-16. Cracks that are apparent in range images but not noticeable in intensity images.

- Raw range image

With the development of laser imaging technology, range image has been adopted by some researchers [29-31,40,42] for surface crack detection. Range-based methods generally rely on the elevation difference in cracked regions to interpret the crack presence. Laser-scanned range images are insensitive to changing illumination condition, and noises such as oil stains and blemishes will not interfere with crack detection on range images [19]. Nevertheless, despite its advantages over intensity images, the range images also have issues such as being sensitive to surface variations and non-crack patterns such as pavement grooves. Moreover, it may be challenging for range-based methods to detect shallow cracks [29,31]. The shallow cracks referred in this study are structural cracks which are difficult to be distinguished in the range data. Based on the shallow cracks identified through this study, the change of elevation is usually under 0.5 mm. Although shallow cracks do not pose a major threat to the health condition of the infrastructure, they can reveal the trend of crack evolution and provide necessary information (e.g., location) to promote precautionary measures. Therefore, accurate detection of the shallow cracks is of importance to the health monitoring and condition assessment of the infrastructure.

- Filtered range image

This study utilizes the filter-based technique [29] proposed in section 3.1 to generate filtered range images, addressing the issues of surface variations and grooved patterns in range image. An example of applying this technique for image pre-processing is illustrated in Figure 3-17. The raw range and filtered range image of a roadway surface are displayed in Figure 3-17 (a) and (b), respectively. As can be observed in this figure, the surface variations and grooved patterns are effectively eliminated from the range image surface, while the cracking features are preserved. In this study, through a comparison on the segmentation performance by the same DCNN trained and tested on raw range vs. filtered range image, the effect of image pre-processing can be demonstrated. Meanwhile, it is noted that, although the use of filtered range images may avoid disturbances such as surface variations and grooved patterns, the uncertainties or subjectivities due to image pre-processing cannot be completely avoided. Therefore, it is preferable to directly utilize raw image data for training and testing to improve the robustness of DCNNs against real-world complexities.

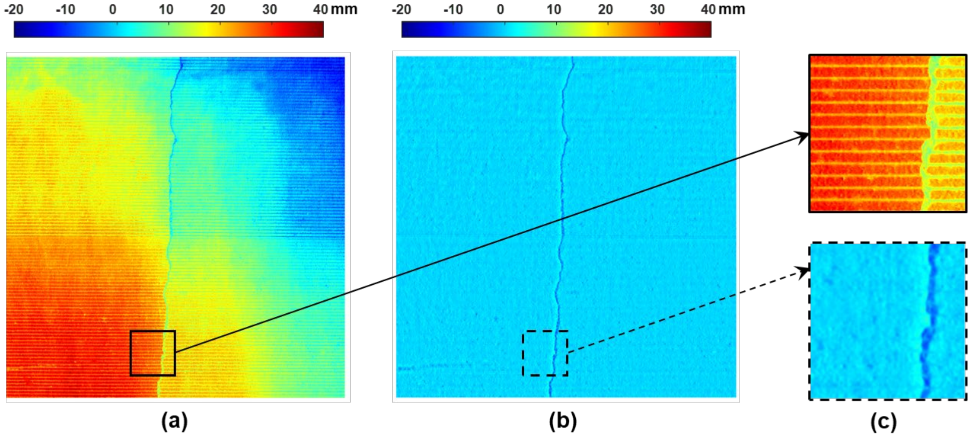


Figure 3-17. An example of image pre-processing: (a) raw range image; (b) filtered range image; and (c) zoomed-in views.

- Data fusion to combine raw intensity and range image

This study investigates the feasibility of directly combining the information in the raw intensity and range image to alleviate issues existing in each type of data. For example, it may occur that cracks of low contrast in intensity images may be more detectable in the corresponding range images; and, in range images, cracks which have shallow depths may be more apparent in the corresponding intensity data. Thus, the use of fused raw image data can provide complementary and more comprehensive information through cross-domain feature correlation and extraction, which may alleviate the issues in individual source of data.

A heterogeneous data fusion strategy is proposed in this study for DCNN-based roadway crack classification and segmentation. As described in section 3.4.1, heterogeneous data fusion is a process to obtain more comprehensive information and reduce uncertainty by integrating multiple sources of data with diverse characteristics instead of examining individual data source [43,132]. In this study, the acquired raw intensity and range images with spatial correspondence are directly fused at data level, by leveraging the concept of hyperspectral imaging [138]. Hyperspectral images have multiple channels with an image component in each channel corresponding to a specific spectral band. In like manner, the raw intensity and range image data can be integrated as hyperspectral imaging. Figure 3-18 illustrates an example of the fused raw image data acquired for crack segmentation, where the channel 1-3 of the hyperspectral image are the RGB channels of a raw range image (dimension: 256×256×3), and the channel 4-6 are the RGB channels of a raw intensity image (dimension: 256×256×3). It is noted that the range and intensity image data is converted into 24-bit RGB image data format during data acquisition. As highlighted in Figure 3-18, the "spatial co-registration" feature is generated through data fusion based on the fact that the intensity and range image data have pixel-to-pixel location correspondence, which is enabled by the 3D laser imaging system (section 2.4). Thus, such features can be exploited to address issues existing in individual data source and facilitate DCNN-based crack segmentation through cross-domain feature correlation and extraction.

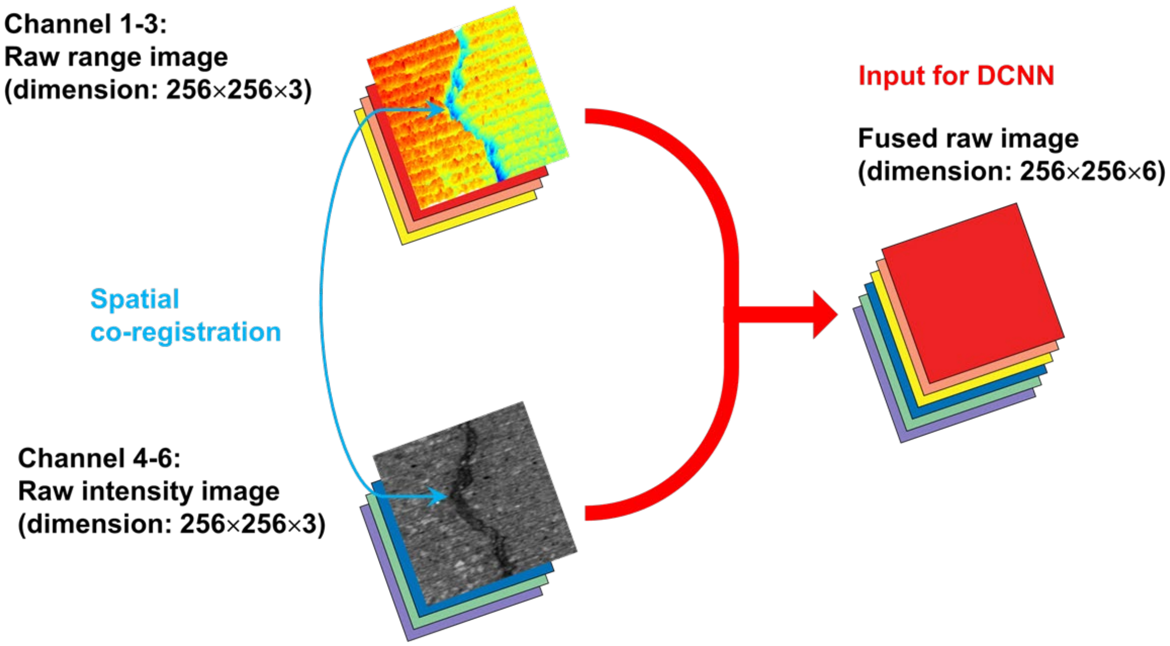


Figure 3-18. An illustration of the fused raw image data.

3.4.3 *Proposed Methodology for Crack Classification*

It is noteworthy that the acquired intensity or range image data used for crack classification by this proposed methodology is 8-bit single-channel data, with the dimension as 256×256×1. Meanwhile, the intensity or range image data used later for crack segmentation is 24-bit three-channel RGB data with a higher resolution, which has the dimension as 256×256×3. Still, the basic concept on the proposed data fusion strategy as illustrated in Figure 3-18 is the same, regardless of the difference in the channel depth of the acquired image data.

In this section, a total of three DCNN architectures are proposed for different tasks. Net-A is designed to take single-channel image input including raw intensity images, raw range images, and filtered range images. Then, Net-B is modified from Net-A by changing the input layer and the first convolutional layer, such that it can utilize fused raw image data for analysis. In addition, another DCNN architecture, Net-C, is proposed as a different layout than Net-B to be trained and tested on the fused raw image data. The proposed architectures include convolutional layers, fully connected layers, and auxiliary layers such as batch normalization, leaky rectified linear unit (LReLU), dropout, and softmax layers, which are introduced in section 2.3.1. It is worth noting that upon designing these architectures, the total number of learnable parameters (see Table 3-5) among these architectures is kept similar to each other, as a means to balance the model complexity for comparison purposes.

- Net-A: a DCNN architecture for single-channel image input

The architecture of Net-A is illustrated in Figure 3-19, with its detailed configuration on each layer tabulated in Table 3-3. This DCNN is a deep architecture which takes single-channel image patches as the input and predicts the images as containing cracks or not.

- Net-B and Net-C: DCNN architectures for dual-channel image input

Two architectures, namely Net-B and Net-C, are configured for fused raw image data. These architectures are designed to have completely different layouts: Net-B directly extracts the spatial co-registration feature in the fused raw image data at a lower level; On the contrary, Net-C first separates the intensity and range information from the input, then performs individual feature extraction and finally merges the extracted features from the intensity or range data at a higher level.

i) Net-B: data fusion at input layer

Having a similar architecture as Net-A, Net-B employs a straightforward yet very intuitive approach to take advantage of the fused raw image data; that is, to modify the kernel sizes of the first convolutional layer ("Conv1" in Table 3-3) from 3×3×1 to 3×3×2 to directly convolve with the dual-channel image input. The detailed layer configuration of Net-B is also tabulated in Table 3-3. The layout of Net-B is illustrated in Figure 3-20. Such a configuration allows to exploit the spatial co-registration features existing in the fused raw image data through the first convolutional layer. Induced by changes in the filter kernels, the total number of parameters is slightly increased from 215572 (Net-A) to 215716 (Net-B).

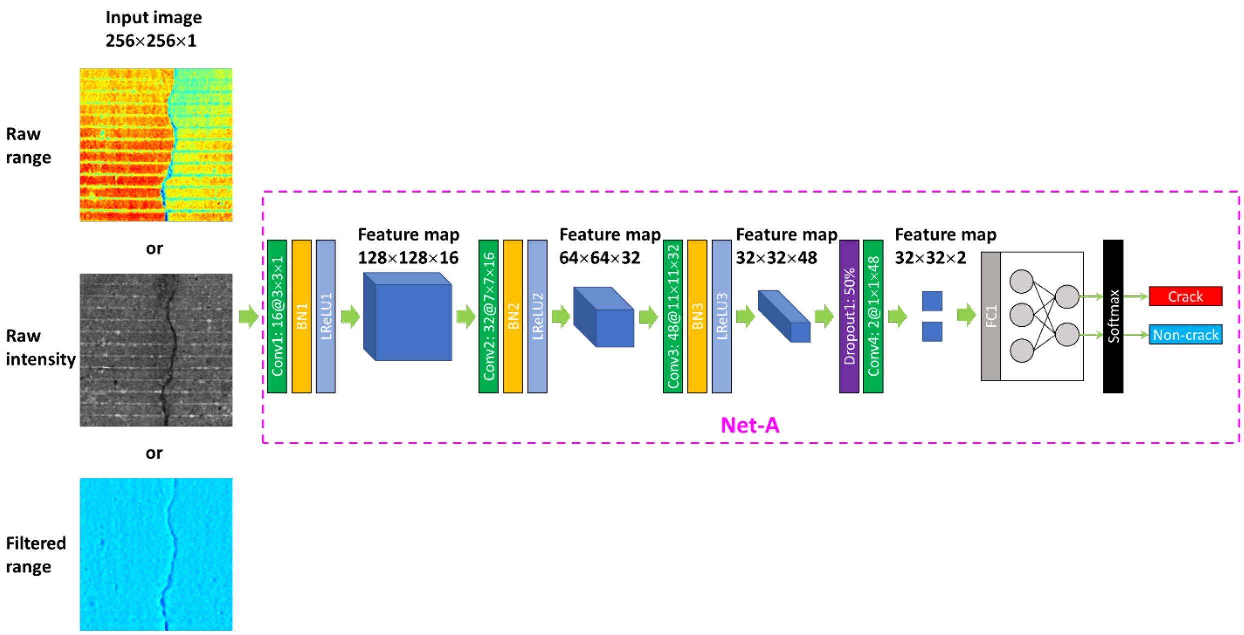


Figure 3-19. Net-A: Proposed DCNN architecture with single-channel image input.

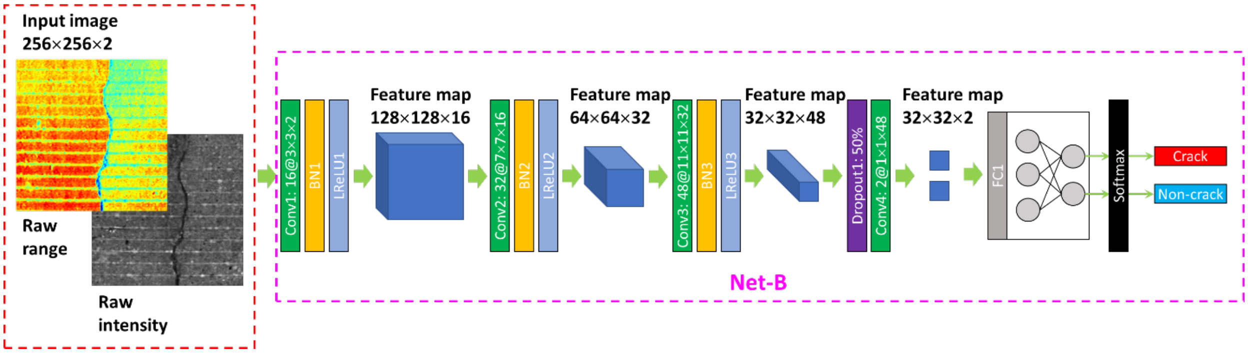


Figure 3-20. Net-B: Proposed DCNN architecture with dual-channel image input.

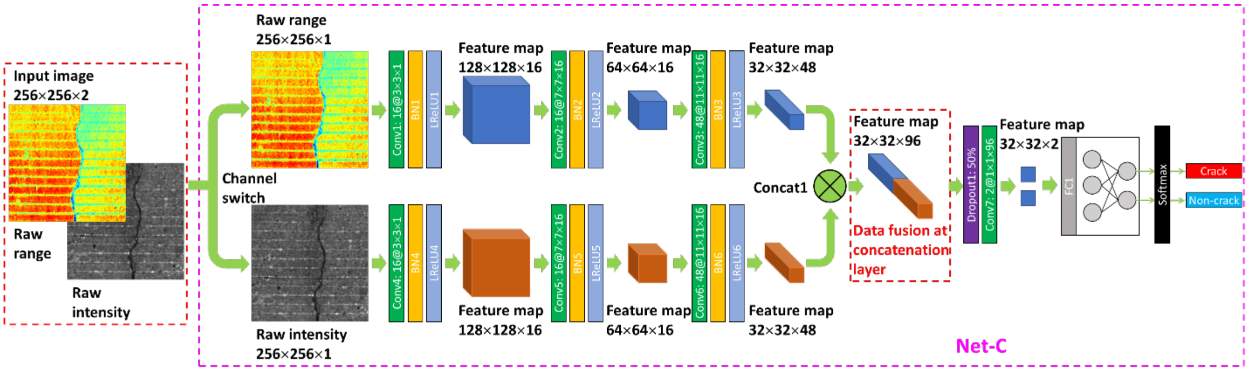


Figure 3-21. Net-C: Proposed DCNN architecture with dual-channel image input.

Table 3-3. Detailed configuration of Net-A and Net-B.

Layer name	Layer type	Output dimension	Kernel size	Depth	Stride	Padding	Learnable parameters			
Input		256×256×1(2)*	-	-	-	-	-			
Conv1	convolution	128×128×16	3×3×1(2)	16	2	1	weight	3×3×1(2)×16	bias	1×16
BN1	batch normalization	128×128×16	-	-	-	-	scale	1×16	shift	1×16
LReLU1	leaky rectified linear unit	128×128×16	-	-	-	-	-			
Conv2	convolution	64×64×32	7×7×16	32	2	3	weight	7×7×16×32	bias	1×32
BN2	batch normalization	64×64×32	-	-	-	-	scale	1×32	shift	1×32
LReLU2	leaky rectified linear unit	64×64×32	-	-	-	-	-			
Conv3	convolution	32×32×48	11×11×32	48	2	5	weight	11×11×32×48	bias	1×48
BN3	batch normalization	32×32×48	-	-	-	-	scale	1×48	shift	1×48
LReLU3	leaky rectified linear unit	32×32×48	-	-	-	-	-			
Dropout1	drop out	32×32×48	-	-	-	-	-			
Conv4	convolution	32×32×2	1×1×48	2	1	0	weight	1×1×48×2	bias	1×2
FC1	full connection	1×2	-	-	-	-	weight	32×32×2×2	bias	1×2
Softmax	softmax normalization	1×2	-	-	-	-	-	-		
							Sum	215572(215716)		

*: the underlined numbers are associated with Net-A, and those in parentheses with Net-B.

ii) Net-C: data fusion at concatenation layer

Different than Net-B, Net-C is designed to perform individual feature extraction on the intensity or range data separated from the fused raw image input, as illustrated in Figure 3-21. The detailed layer configuration is tabulated in Table 3-4. Separation on the input data is achieved by convolving the fused raw image input with a fixed-weight 1×1 convolutional layer, which acts like a channel switch. After feature extraction on separate channels, the high-level information is merged through a depth concatenation layer, labeled as "Concat1" in Figure 3-21 and Table 3-4. The major difference between Net-B and Net-C lies in that Net-B fuses image information at a lower level through a convolutional layer, referred to as a "fuse-extract" pattern; Net-C, however, fuses features extracted from separate image channels at a higher level through a concatenation (i.e., fusion) layer, which can be referred to as a "extract-fuse" pattern. As demonstrated in Case II, different patterns on the data fusion and feature extraction govern the network performance.

As mentioned previously, these DCNN architectures are designed such that they contain similar amounts of learnable parameters, as indicated in Table 3-5. With all three architectures sharing the similar level of model complexity as reflected by the number of parameters, the major impact factors on the network performance thus originate from different types of image data and the associated architecture layouts.

- Flow chart of the proposed DCNN-based crack classification methodology with heterogeneous image fusion

The flow chart of the proposed DCNN-based crack classification methodology based on heterogeneous image fusion is illustrated in Figure 3-22. As shown in this figure, the first step is to obtain raw range and intensity image data by using the laser imaging system (section 2.4). Second, the proposed heterogeneous image fusion approach (section 3.4.2) is employed to integrate the raw range and intensity data. Then, the fused raw image is processed through a data generation process, including image patch generation through the sliding window technique (section 2.3.2), ground truth label (i.e., "crack" vs. "non-crack") generation, data augmentation, and training/validation datasets generation. The next two steps are to train the proposed DCNN architectures on the fused image data and then utilize the trained DCNNs to predict a crack map on a new roadway image data.

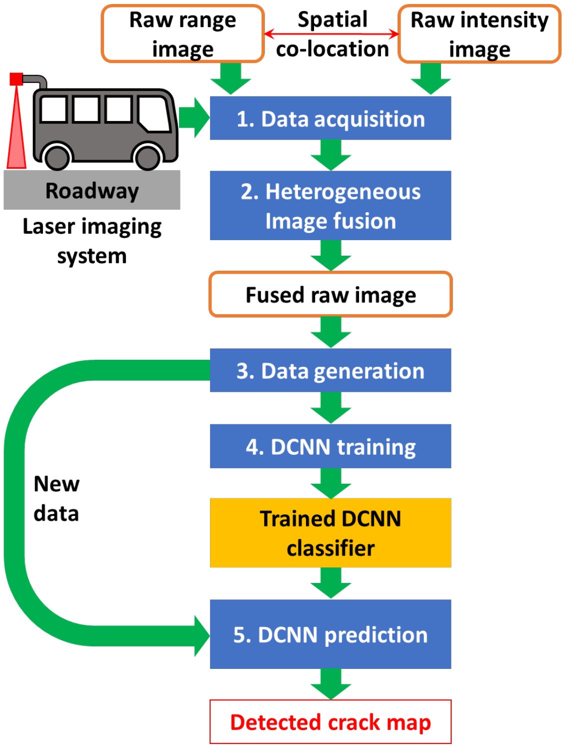


Figure 3-22. Flow chart of the proposed DCNN-based crack classification methodology with heterogeneous image fusion.

Note that the flow chart as shown in Figure 3-22 refers to the crack classification tasks based on heterogeneous image fusion. For DCNN-based crack segmentation tasks with heterogeneous image fusion, which will be introduced in section 3.4.4, they share the same flow chart except that the DCNNs are developed for segmentation instead of classification.

Table 3-4. Detailed configuration of Net-C.

Layer name	Layer type	Output dimension	Kernel size	Depth	Stride	Padding	Learnable parameters			
Input		256×256×2	-	-	-	-	-			
Conv1	convolution	128×128×16	3×3×1	16	2	1	weight	3×3×1×16	bias	1×16
BN1	batch normalization	128×128×16	-	-	-	-	scale	1×16	shift	1×16
LReLU1	leaky rectified linear unit	128×128×16	-	-	-	-	-			
Conv2	convolution	64×64×16	7×7×16	16	2	3	weight	7×7×16×16	bias	1×16
BN2	batch normalization	64×64×16	-	-	-	-	scale	1×16	shift	1×16
LReLU2	leaky rectified linear unit	64×64×16	-	-	-	-	-			
Conv3	convolution	32×32×48	11×11×16	48	2	5	weight	11×11×16×48	bias	1×48
BN3	batch normalization	32×32×48	-	-	-	-	scale	1×48	shift	1×48
LReLU3	leaky rectified linear unit	32×32×48	-	-	-	-	-			
Conv4	convolution	128×128×16	3×3×1	16	2	1	weight	3×3×1×16	bias	1×16
BN4	batch normalization	128×128×16	-	-	-	-	scale	1×16	shift	1×16
LReLU4	leaky rectified linear unit	128×128×16	-	-	-	-	-			
Conv5	convolution	64×64×16	7×7×16	16	2	3	weight	7×7×16×16	bias	1×16
BN5	batch normalization	64×64×16	-	-	-	-	scale	1×16	shift	1×16
LReLU5	leaky rectified linear unit	64×64×16	-	-	-	-	-			
Conv6	convolution	32×32×48	11×11×16	48	2	5	weight	11×11×16×48	bias	1×48
BN6	batch normalization	32×32×48	-	-	-	-	scale	1×48	shift	1×48
LReLU6	leaky rectified linear unit	32×32×48	-	-	-	-	-			
Concat1	concatenation	32×32×96	-	-	-	-	-			
Dropout1	drop out	32×32×96	-	-	-	-	-			
Conv7	convolution	32×32×2	1×1×96	2	1	0	weight	1×1×96×2	bias	1×2
FC1	full connection	1×2	-	-	-	-	weight	32×32×2×2	bias	1×2
Softmax	softmax normalization	1×2	-	-	-	-	-	-		
							Sum	216004		

Table 3-5. Number of parameters of the proposed architectures.

Architecture	Data type	Number of parameters
Net-A	raw intensity, raw range, filtered range	215572
Net-B	fused raw image	215716
Net-C	fused raw image	216004

3.4.4 *Proposed Methodology for Crack Segmentation*

As explained previously, the intensity or range image data acquired for crack segmentation is 24-bit three-channel RGB data with the dimension as 256×256×3, which has a higher resolution than the image data used for crack classification (i.e., 8-bit single-channel image data).

Three encoder-decoder networks, denoted as Net-1, 2, and 3, are proposed in this study to exploit heterogeneous image data. It is noteworthy that these architectures are designed such that they consume similar amounts of parameters indicating their similar model complexity.

- Net-1: An encoder-decoder network for a single type of image data

Figure 3-23 illustrates the layout of Net-1, an encoder-decoder network designed to take a single type of image data, such as raw intensity, raw range, or filtered range image. As displayed in Figure 3-23, the encoder of Net-1 contains five convolutional blocks. Each convolutional block consists of a convolutional layer using multiple 3×3 kernels with a stride of 2 for feature extraction, a batch normalization layer to improve generalization, and a LReLU layer to provide nonlinearity. After feature extraction through the encoder, a convolutional layer with 1×1 kernels is utilized for cross-channel pooling. Subsequently, in the decoder, five transposed convolutional blocks are adopted. Each transposed convolutional block consists of a transposed convolutional layer for feature up-sampling, and auxiliary layers including batch normalization and LReLU. At the end of the decoder, a convolutional layer is utilized for cross-channel pooling on the expanded feature maps; and, the output is normalized by a softmax layer to generate a crack probability map. Detailed architecture configuration is tabulated in Table 3-6.

It is noted that the convolutional and transposed convolutional blocks are connected through residual connections, as illustrated in Figure 3-23. Accordingly to [82,101], adding residual connections in a deep architecture can help prevent performance degradation and avoid data singularity. In this study, low-level features extracted by the convolutional blocks in the encoder are merged with the high-level feature output of the transposed convolutional blocks in the decoder through an arithmetic addition operation.

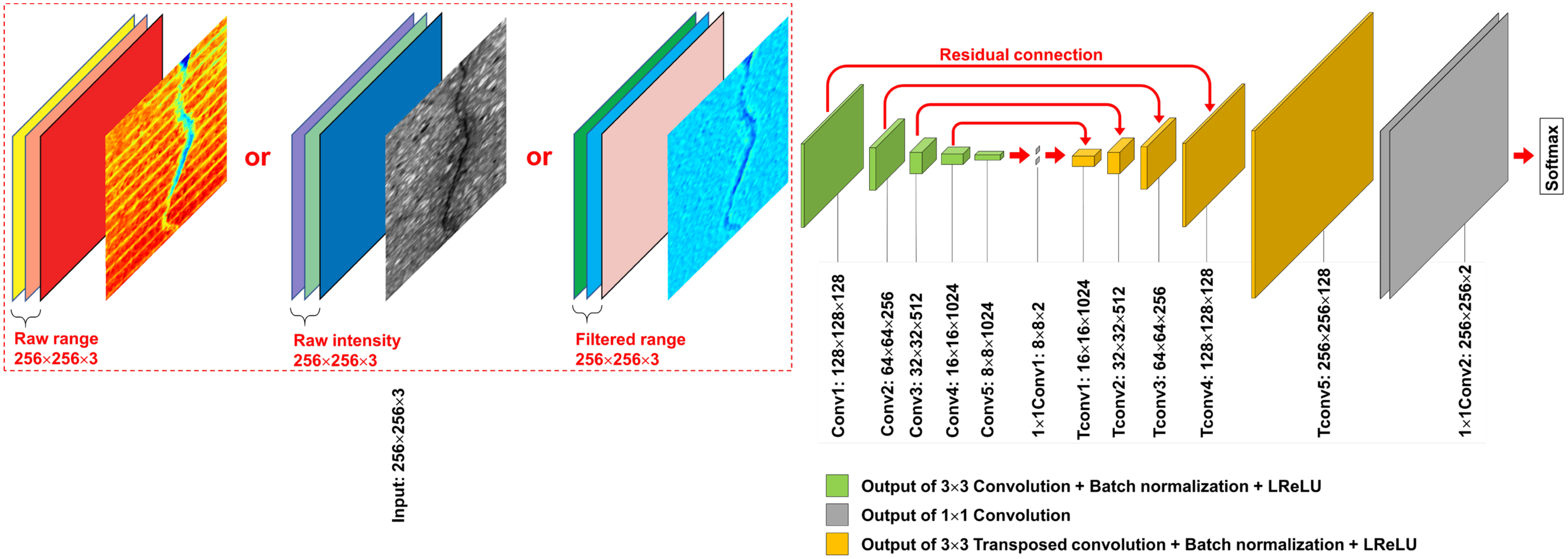


Figure 3-23. Net-1: An encoder-decoder network to take a single type of image data.

Table 3-6. Layer configurations of Net-1 and Net-2.

Layer name	Layer type**	Output dimension	Kernel size	Depth	Stride	Padding	Cropping	Learnable parameters			
Input	INPUT	256×256×3(6)*	-	-	-	-	-	-			
Conv1	CONV	128×128×128	3×3×3(6)	128	2	1	-	weight	3×3×3(6)×128	bias	1×128
	BN	128×128×128	-	-	-	-	-	scale	1×128	shift	1×128
	LReLU	128×128×128	-	-	-	-	-	-			
Conv2	CONV	64×64×256	3×3×128	256	2	1	-	weight	3×3×128×256	bias	1×256
	BN	64×64×256	-	-	-	-	-	scale	1×256	shift	1×256
	LReLU	64×64×256	-	-	-	-	-	-			
Conv3	CONV	32×32×512	3×3×256	512	2	1	-	weight	3×3×256×512	bias	1×512
	BN	32×32×512	-	-	-	-	-	scale	1×512	shift	1×512
	LReLU	32×32×512	-	-	-	-	-	-			
Conv4	CONV	16×16×1024	3×3×512	1024	2	1	-	weight	3×3×512×1024	bias	1×1024
	BN	16×16×1024	-	-	-	-	-	scale	1×1024	shift	1×1024
	LReLU	16×16×1024	-	-	-	-	-	-			
Conv5	CONV	8×8×1024	3×3×1024	1024	2	1	-	weight	3×3×1024×1024	bias	1×1024
	BN	8×8×1024	-	-	-	-	-	scale	1×1024	shift	1×1024
	LReLU	8×8×1024	-	-	-	-	-	-			
1×1Conv1	CONV	8×8×2	1×1×1024	2	1	0	-	weight	1×1×1024×2	bias	1×2
Tconv1	TransCONV	16×16×1024	3×3×2	1024	2	-	1	weight	3×3×2×1024	bias	1×1024
	BN	16×16×1024	-	-	-	-	-	scale	1×1024	shift	1×1024
	LReLU	16×16×1024	-	-	-	-	-	-			
Tconv2	TransCONV	32×32×512	3×3×1024	512	2	-	1	weight	3×3×1024×512	bias	1×512
	BN	32×32×512	-	-	-	-	-	scale	1×512	shift	1×512
	LReLU	32×32×512	-	-	-	-	-	-			
Tconv3	TransCONV	64×64×256	3×3×512	256	2	-	1	weight	3×3×512×256	bias	1×256
	BN	64×64×256	-	-	-	-	-	scale	1×256	shift	1×256
	LReLU	64×64×256	-	-	-	-	-	-			
Tconv4	TransCONV	128×128×128	3×3×256	128	2	-	1	weight	3×3×256×128	bias	1×128
	BN	128×128×128	-	-	-	-	-	scale	1×128	shift	1×128
	LReLU	128×128×128	-	-	-	-	-	-			
Tconv5	TransCONV	256×256×128	3×3×128	128	2	-	1	weight	3×3×128×128	bias	1×128
	BN	256×256×128	-	-	-	-	-	scale	1×128	shift	1×128
	LReLU	256×256×128	-	-	-	-	-	-			
1×1Conv2	CONV	256×256×2	1×1×128	2	1	0	-	weight	1×1×128×2	bias	1×2
Softmax	SOFTMAX	1×2	-	-	-	-	-	-			
								Sum	**22010116** (22013572)		

*: The numbers in parentheses are associated with Net-2.

**: INPUT: image input layer; CONV: convolutional layer; BN: batch normalization layer; LReLU: leaky rectified linear unit; TransCONV: transposed convolutional layer; SOFTMAX: softmax normalization layer.

- Net-2 and Net-3: Encoder-decoder networks for fused raw image data

Two encoder-decoder networks, Net-2 and 3, are developed for fused raw image data. The major difference is that Net-2 directly exploits the fused raw image data containing spatial co-registration features through a convolution operation, which can be referred to as a “fuse-extract” pattern; Net-3, which has a two-stream encoder layout, performs feature extraction on separate image data, and then fuse the high-level features through an addition operation, which can be considered as an “extract-fuse” pattern. Net-2 and 3 have the same decoder layout, hence the influencing factor on their segmentation performance stems from different strategies to exploit the fused raw image data and the associated encoder layouts.

i) Net-2: An encoder-decoder network with a “fuse-extract” pattern

Net-2 represents a straightforward and intuitive approach by modifying the input layer of a DCNN to exploit the fused raw image data. As illustrated in Figure 3-24, the layout of Net-2 is the same as that of Net-1 except the image input layer, where the input for Net-2 contains 6 channels comprised of raw range and intensity data. The layer configuration of Net-2 is also tabulated in Table 3-6. Due to the change of input dimension, the kernel sizes in the first convolutional layer (see “Conv1” in Table 3-6) are changed accordingly. And, the total number of parameters is increased from 22010116 (Net-1) to 22013572 (Net-2).

ii) Net-3: An encoder-decoder network with an “extract-fuse” pattern

As illustrated in Figure 3-25, the encoder of Net-3 contains two streams, one for the raw range image component, the other for the raw intensity image component. In the encoder, feature extraction is performed separately on the raw range or intensity image data. The extracted features in each stream are then fused through an addition layer (“Add1” in Figure 3-25 and Table 3-7), which performs an arithmetic addition operation. Such a fusion strategy can be considered as an “extract-fuse” pattern, different than the “fuse-extract” pattern adopted in Net-2. Regarding the decoder, Net-3 utilizes the same decoder module as Net-1 and 2 for feature map expansion. Meanwhile, residual connections are employed to merge the low-level features from both streams in the encoder with the high-level features in the decoder. The layer configuration is tabulated in Table 3-7. It is noted that, the total number of parameters of Net-3 (23491334) is similar to that of Net-2 (22013572), implying very similar model complexity. Then, the difference in segmentation performance on the fused raw image data is solely induced by the different encoder modules.

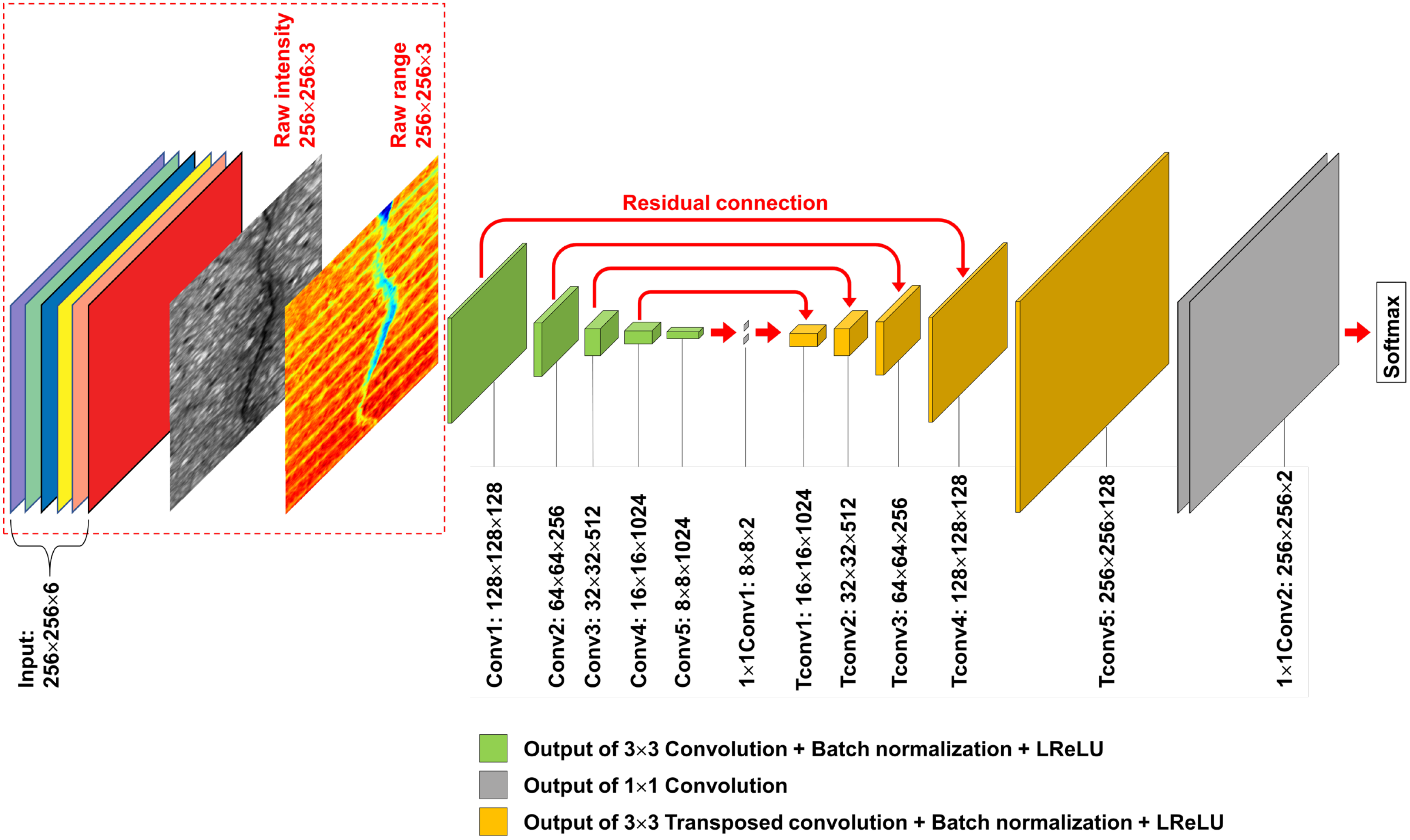


Figure 3-24. Net-2: An encoder-decoder network to take fused raw image data.

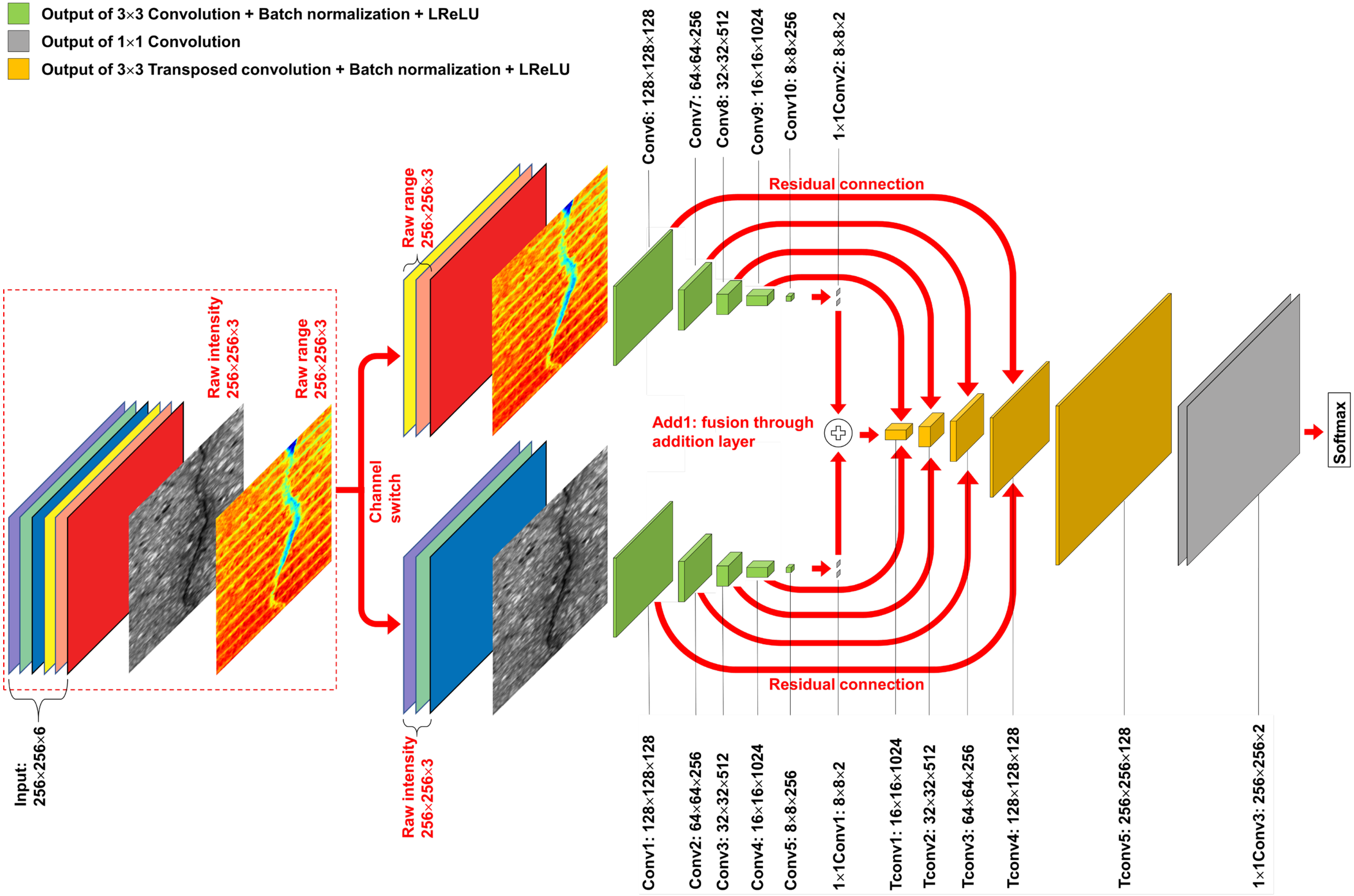


Figure 3-25. Net-3: A two-stream encoder-decoder network to take fused raw image data.

Table 3-7. Layer configuration of Net-3.

Layer name	Layer type*	Output dimension	Kernel size	Depth	Stride	Padding	Cropping	Learnable parameters			
Input	INPUT	256×256×6	-	-	-	-	-	-			
Conv1 /Conv6	CONV	128×128×128	3×3×3	128	2	1	-	weight	3×3×3×128	bias	1×128
	BN	128×128×128	-	-	-	-	-	scale	1×128	shift	1×128
	LReLU	128×128×128	-	-	-	-	-	-			
Conv2 /Conv7	CONV	64×64×256	3×3×128	256	2	1	-	weight	3×3×128×256	bias	1×256
	BN	64×64×256	-	-	-	-	-	scale	1×256	shift	1×256
	LReLU	64×64×256	-	-	-	-	-	-			
Conv3 /Conv8	CONV	32×32×512	3×3×256	512	2	1	-	weight	3×3×256×512	bias	1×512
	BN	32×32×512	-	-	-	-	-	scale	1×512	shift	1×512
	LReLU	32×32×512	-	-	-	-	-	-			
Conv4 /Conv9	CONV	16×16×1024	3×3×512	1024	2	1	-	weight	3×3×512×1024	bias	1×1024
	BN	16×16×1024	-	-	-	-	-	scale	1×1024	shift	1×1024
	LReLU	16×16×1024	-	-	-	-	-	-			
Conv5 /Conv10	CONV	8×8×256	3×3×1024	256	2	1	-	weight	3×3×1024×256	bias	1×256
	BN	8×8×256	-	-	-	-	-	scale	1×256	shift	1×256
	LReLU	8×8×256	-	-	-	-	-	-			
1×1Conv1 /1×1Conv2	CONV	8×8×2	1×1×256	2	1	0	-	weight	1×1×256×2	bias	1×2
Add1	Addition layer	8×8×2	-	-	-	-	-	-			
Tconv1	TransCONV	16×16×1024	3×3×2	1024	2	-	1	weight	3×3×2×1024	bias	1×1024
	BN	16×16×1024	-	-	-	-	-	scale	1×1024	shift	1×1024
	LReLU	16×16×1024	-	-	-	-	-	-			
Tconv2	TransCONV	32×32×512	3×3×1024	512	2	-	1	weight	3×3×1024×512	bias	1×512
	BN	32×32×512	-	-	-	-	-	scale	1×512	shift	1×512
	LReLU	32×32×512	-	-	-	-	-	-			
Tconv3	TransCONV	64×64×256	3×3×512	256	2	-	1	weight	3×3×512×256	bias	1×256
	BN	64×64×256	-	-	-	-	-	scale	1×256	shift	1×256
	LReLU	64×64×256	-	-	-	-	-	-			
Tconv4	TransCONV	128×128×128	3×3×256	128	2	-	1	weight	3×3×256×128	bias	1×128
	BN	128×128×128	-	-	-	-	-	scale	1×128	shift	1×128
	LReLU	128×128×128	-	-	-	-	-	-			
Tconv5	TransCONV	256×256×128	3×3×128	128	2	-	1	weight	3×3×128×128	bias	1×128
	BN	256×256×128	-	-	-	-	-	scale	1×128	shift	1×128
	LReLU	256×256×128	-	-	-	-	-	-			
1×1Conv3	CONV	256×256×2	1×1×128	2	1	0	-	weight	1×1×128×2	bias	1×2
Softmax	SOFTMAX	1×2	-	-	-	-	-	-			
								Sum	23491334		

*: same as defined in Table 3-6.

CHAPTER 4: EXPERIMENTAL STUDY AND RESULTS

The proposed crack detection methodologies are applied to image data of real-world complexities. And, the corresponding experimental results and discussions are presented in this chapter.

4.1 Image Processing Technique for Robust Crack Detection Using Range Image Data

This section presents an experimental study of applying the proposed crack detection methodology on bridge deck surfaces. The intensity and range image data of a bridge deck are acquired by the laser imaging system as introduced in section 2.4. The range data has the resolution of $p_u = 2$ mm/pixel and $p_v = 1$ mm/pixel along longitudinal and transverse direction, respectively. Three images containing cracks with different noise level, average elevation, and crack pattern, are selected from the data for illustration purpose. The same frequency domain filters are applied to the three bridge deck range images, and the results of image filtering are demonstrated in section 4.1.1. The next section 4.1.2 shows the crack detection results on the filtered surfaces using the contouring analysis. Then, in section 4.1.3, the accuracy of the proposed crack detection methodology is demonstrated through a precision-recall analysis. The limitation of the proposed technique is summarized in section 4.1.4.

4.1.1 *Image Pre-Processing using Frequency Domain Filtering*

Prior to detecting cracks, image pre-processing is conducted to reduce the influence from grooved patterns and surface variations (Figure 3-1) such as ripples and rutting, and meanwhile suppress the image noises. During the pre-processing, three filtering processes are performed sequentially. First, a high-pass filtering is applied to the bridge deck range data, to remove surface variations; then, a low-pass filtering is performed on the high-pass filtered surface for noise suppression; finally, a multiple notch filtering is conducted to remove grooved patterns in the surface.

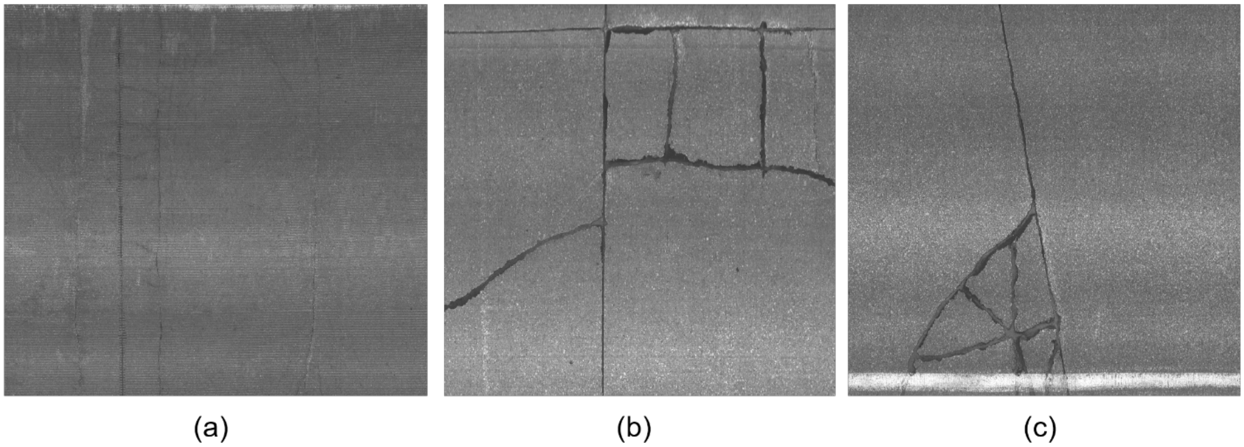


Figure 4-1. Intensity images of bridge deck surfaces: (a) Surface 1; (b) Surface 2; and (c) Surface 3.

The intensity images and range images of the bridge deck surfaces are shown in Figure 4-1 and Figure 4-2 (a-c), respectively. In both figures, the horizontal axis indicates the longitudinal vehicle driving direction. In Surface 1 [Figure 4-1 (a) and Figure 4-2 (a)], both longitudinal grooved patterns and thin cracks propagating along the transverse direction exist; Surface 2 [Figure 4-1 (b) and Figure 4-2 (b)] does not have grooves, but contains a punchout cracking pattern; in Surface 3 [Figure 4-1 (c) and Figure 4-2 (c)], a corner break cracking pattern can be observed. Figure 4-1 (a-c) show the conventional intensity images, and Figure 4-2 (a-c) illustrate the original range surfaces. In all three range images, the color scale is associated with surface elevation. Judging from the color change, significant surface variations can be observed, especially in Surface 2.

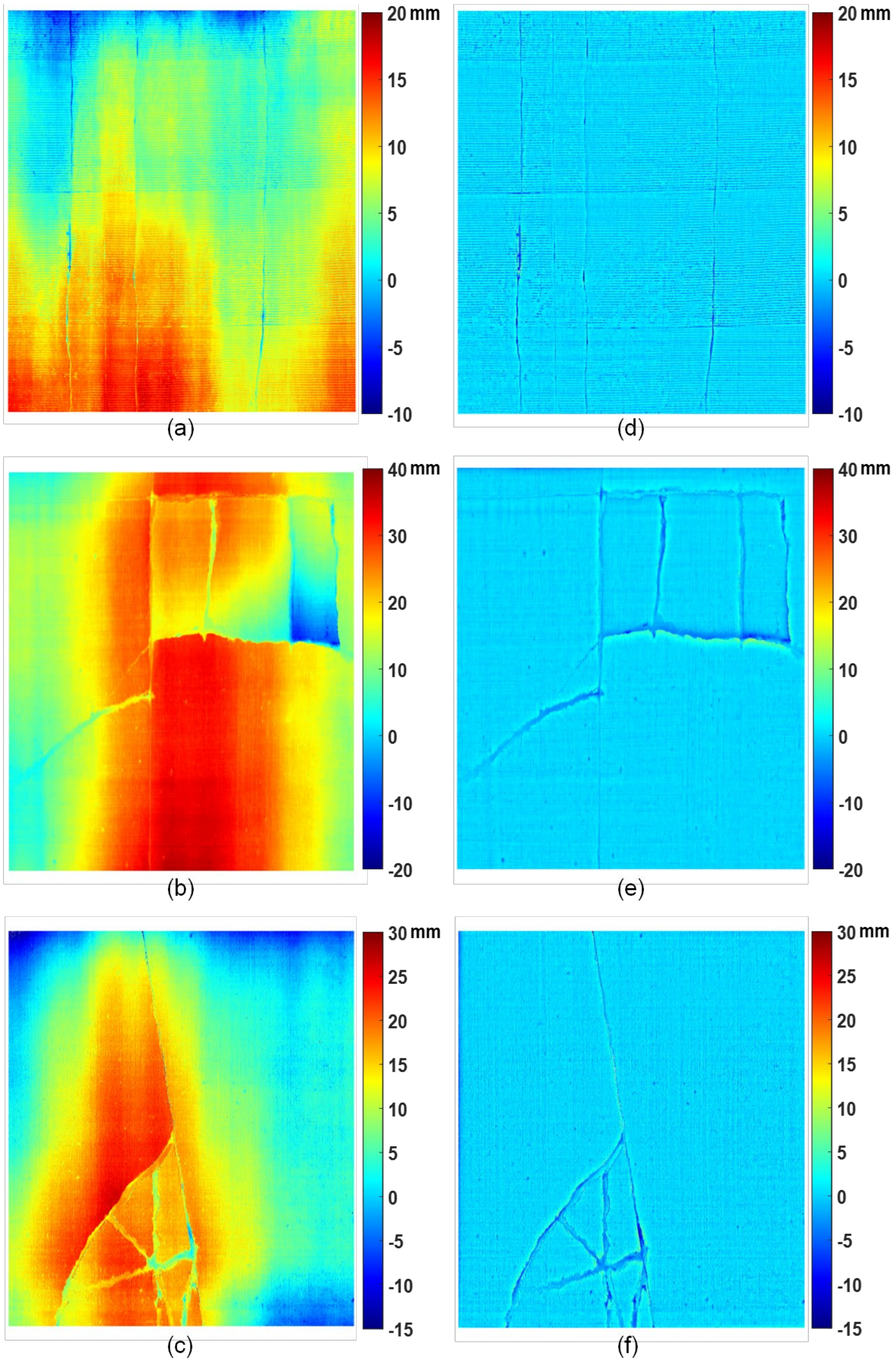


Figure 4-2. Original range surfaces and the high-pass filtering result: (a-c) the original range Surface 1, 2, and 3; and (d-f) the high-pass filtered range Surface 1, 2, and 3.

- High-pass filtering result

A Gaussian high-pass filter is applied to the original range surfaces to remove surface variations. The maximum crack width to be detected is selected as 200 mm—a threshold smaller than 200 mm can also be chosen, but a larger maximum crack width is generally considered as a more challenging case as more cracks are preserved in the analysis. Then, based on the relationship between the width of a crack and its frequency domain information derived in section 3.1.4, the high-pass cutoff frequency is equal to $\frac{1}{200}$ cycle/mm. The corresponding high-pass filtering results are demonstrated in Figure 4-2 (d-f). In all three filtered surfaces, the surface color becomes monochromatic in blue, which corresponds to the color representing zero elevation. This result indicates that, the Gaussian high-pass filter can effectively remove the surface variations. More importantly, as can be also observed in the filtered images, cracks are still preserved through this filtering process. This filtering result justifies the choice of the cutoff frequency based on the crack width.

- Low-pass filtering result

As a subsequent step, a Gaussian low-pass filter is applied to the high-pass filtered surfaces for noise suppression purpose. Generally, the higher the resolution of the data, the smaller the minimum crack width can be detected. During the low-pass filtering process in this study, the minimum width of the preserved cracks is selected as 10 mm. Therefore, according to section 3.1.4, the low-pass cutoff frequency is equal to $\frac{1}{10}$ cycle/mm. Results of the low-pass filtering are demonstrated in Figure 4-3. For each range surface, the side views (along the longitudinal vehicle driving direction) of the original surface [Figure 4-3 (a-c)], high-pass filtered surface [Figure 4-3 (d-f)], and low-pass filtered surface [Figure 4-3 (g-i)] are illustrated in this figure, respectively. Again, by comparing the side views of the original surface [Figure 4-3 (a-c)] and high-pass filtered surface [Figure 4-3 (d-f)], it is clear that the high-pass filter is capable of removing the surface variations, and the filtered surface is centered at zero elevation. Meanwhile, by applying the proposed low-pass filtering, the image noises are suppressed, as can be clearly observed in Figure 4-3 (g) and (i). Judging from the elevation information, the transverse cracks in Surface 1, indicated in Figure 4-3 (g), are preserved through the low-pass filtering process. Through these low-pass filtering examples, the effectiveness of the Gaussian low-pass filter on noise removal is successfully demonstrated.

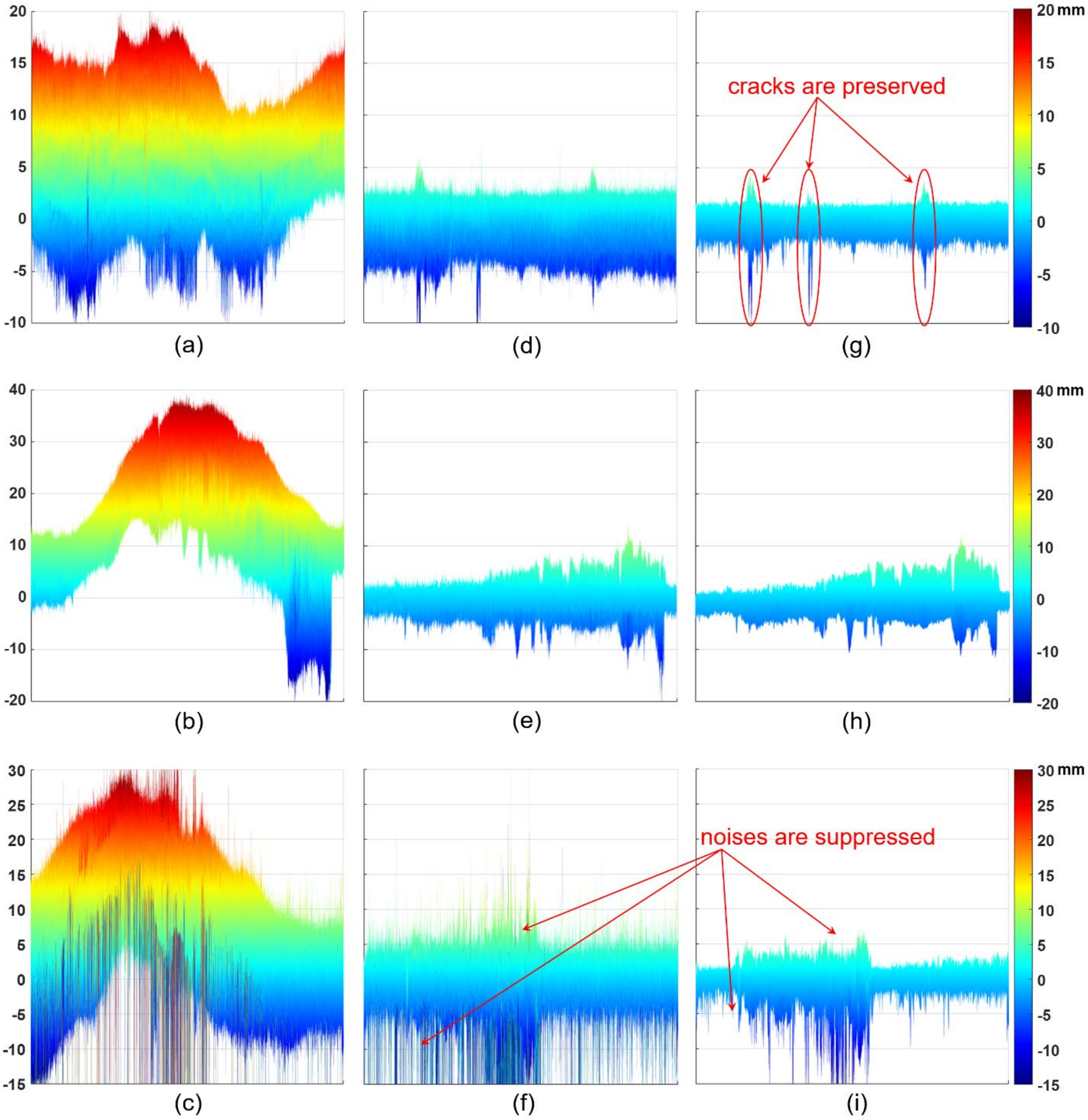


Figure 4-3. Low-pass filtering result: side views of (a-c) the original range Surface 1, 2, and 3; (d-f) the high-pass filtered range Surface 1, 2, and 3; and (g-i) the high-pass and low-pass filtered range Surface 1, 2, and 3.

- Multiple notch filtering result

The grooved patterns existing in Surface 1 can be removed by using the proposed notch filtering technique. Based on the information presented in section 3.1.4, the following parameters are used to design the multiple notch filter (see Figure 3-7): groove spacing $g = 20$ mm; orientation angle $\theta = 0°$; number of harmonics to be filtered out $Q = 4$; frequency radius $D_n = 0.2F_0$. Result of the notch filtering is illustrated in Figure 4-4. By applying the notch filtering, the grooved patterns are successfully removed from the surface, as can be seen from the zoomed-in view in Figure 4-4 (b) and (c) for a comparison on before and after filtering. It is evident that the proposed notch filtering technique is capable of effectively removing the grooved patterns while preserving the cracks.

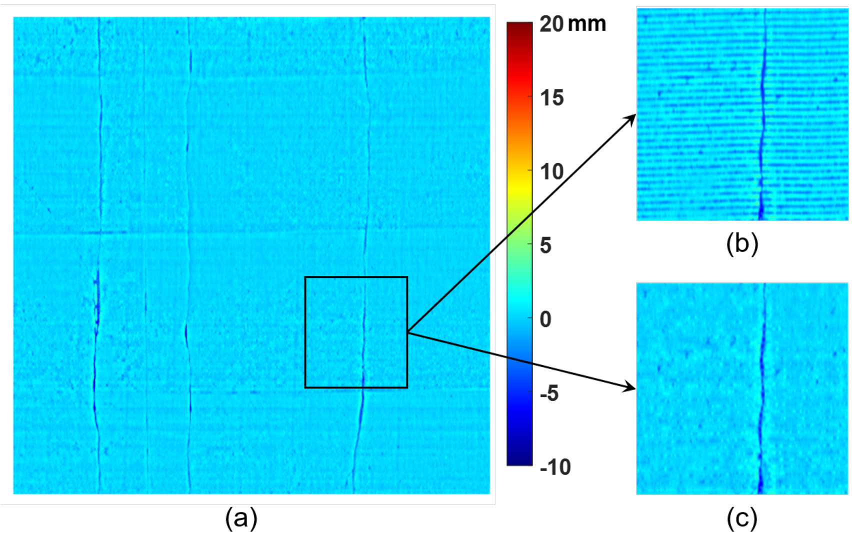


Figure 4-4. Multiple notch filtering result on Surface 1: (a) the high-pass, low-pass, and notch filtered range surface; (b) zoomed-in view before notch filtering; and (c) zoom-in view after notch filtering.

4.1.2 *Crack Detection Results Based on Contouring Analysis*

Through the filtering processes, both the surface variations and grooved patterns are eliminated from the surface, and the image noises are significantly reduced. Subsequently, the contouring analysis implemented using marching squares algorithm and associated parameters (section 3.1.5) is applied on the filtered image surface, and the obtained contour maps are displayed in Figure 4-5 (a-c). This figure shows the extracted contours by thresholding each filtered range surface at $h = -1$ mm. In this figure, the cracks are extracted from all three surfaces, but meanwhile many small contours are also detected due to the remnant image noises.

By applying the criteria expressed in Equations (3-8) and (3-9), majority of the non-crack contours can be removed. In this study, the parameters in Equations (3-8) and (3-9) are assigned as: $tol_1 = 100$ mm^2, and $tol_2 = 0.2$. After removing the non-crack contours, the cracks are extracted from the contour maps, and superimposed on the filtered range images, as shown in Figure 4-5 (d-f). These filtered range images are considered as the ground truth. In Surface 1 [Figure 4-5 (d)], the transverse cracks are clearly extracted; in Surfaces 2 [Figure 4-5 (e)] and 3 [Figure 4-5 (f)], the punchout and corner break cracking patterns are detected with high accuracy as well, as can be observed from the zoomed-in view in Figure 4-5 (h) and (i). By comparing each contour map with the corresponding detected cracks, it can be seen that the proposed non-crack contour removal criteria [Equations (3-8) and (3-9)] are very effective on discriminating the real cracks from the non-crack contours.

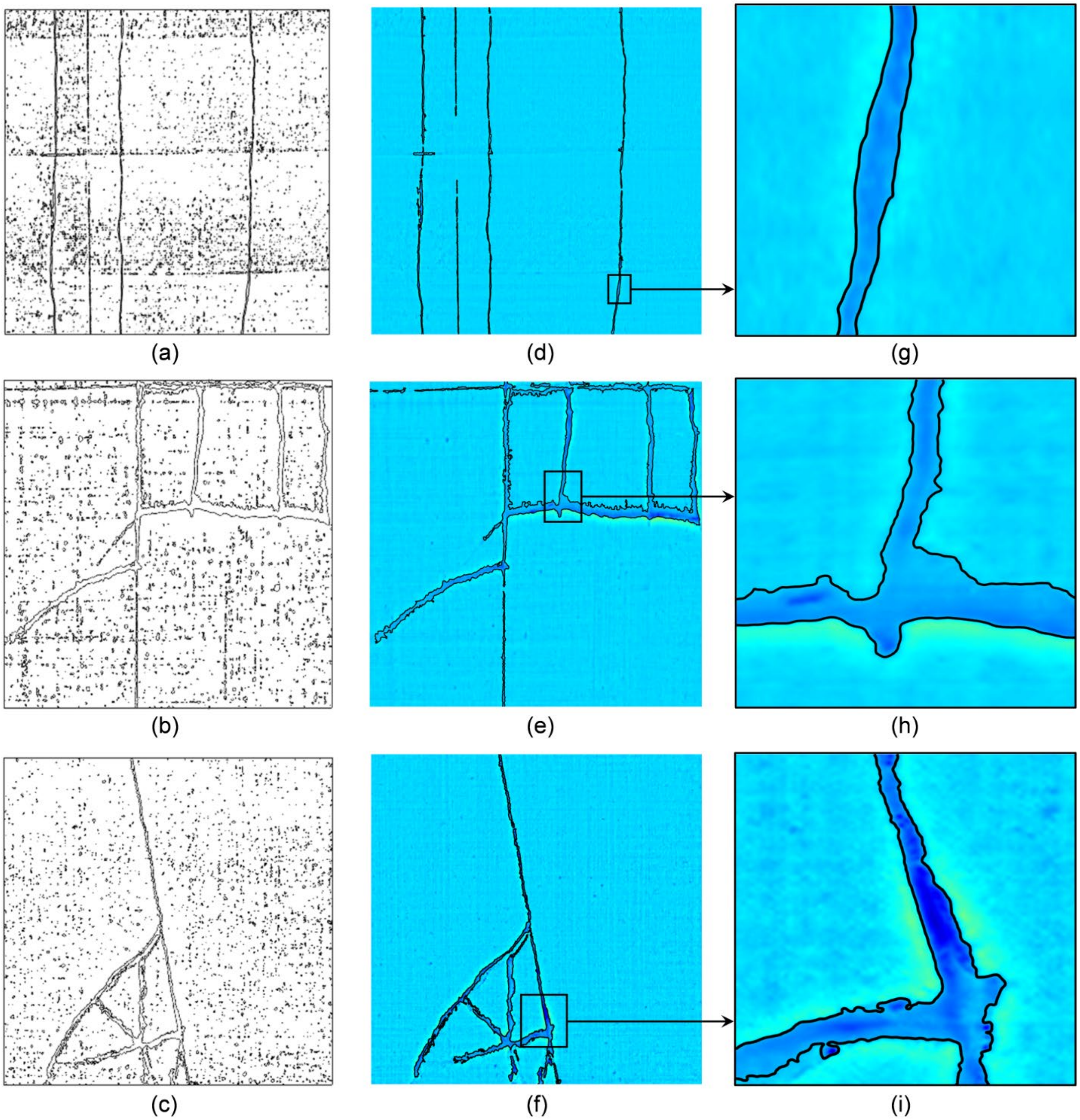


Figure 4-5. Detected contours and cracks: (a-c) contour detection results on the filtered range Surface 1, 2, and 3; (d-f) crack detection results on the filtered range Surface 1, 2, and 3; and (g-i) zoomed-in views of the crack detection results.

4.1.3 *Validation*

A validation procedure is performed using the precision-recall analysis (section 2.5.1). In this study, the ground truth crack pixels are obtained through a binarization procedure on the filtered range images followed by a manual pixel selection process. The result of the precision-recall analysis is summarized in Table 4-1. The performance metrics are calculated using the crack detection results of all three images considered in the experimental study. As presented in Table 4-1, the average values of the Precision, Recall, and F1 are 91.7%, 97.1%, and 94.3%, respectively, indicating high accuracy of the proposed crack detection methodology. Moreover, judging by these metrics, the proposed methodology can achieve consistent accuracy on detection of different types of cracks under the influence of grooved patterns, elevation variations, and image noises, thus demonstrating the robustness of the proposed methodology.

In addition, the precision-recall analysis is performed on the crack detection results of the first image [Figure 4-2 (a)] under three scenarios: *i*) only the high-pass filtering is not applied in the pre-processing; *ii*) only the low-pass filtering is not applied; and *iii*) only the notch filtering is not applied. The resulted F1 values in the three scenarios are 13.0%, 74.2%, and 18.3%, respectively. The rapidly deteriorated values demonstrate poor crack detection performance when one of the filtering processes is missing, indicating that the proposed methodology is a systematic integral framework and each component is necessary to provide robust performance.

To further validate the proposed crack detection methodology, a comparison is performed between the proposed methodology and a seed-based crack detection methodology [28]. This technique was extended from grid cell analysis, a fast and accurate crack detection technique that was first proposed by [51] and later used and extended by many other researchers [28,53,139-141] for crack detection and comparison purposes.

The seed-based approach is performed on the same filtered image data and evaluated using the same performance metrics as the proposed approach. The results are presented in Table 4-1. The binary maps of *i*) the ground truth, *ii*) the cracks detected by the proposed methodology, and *iii*) by the seed-based approach on Surface 1 are provided in Figure 4-6 as an illustrative example. As can be observed from this figure, both the proposed methodology and the seed-based approach can extract the major trend of the cracks, but the latter failed to detect some cracks (i.e., false negative). From Table 4-1, the average Precision of the proposed methodology is similar to that of the seed-based approach; however, the average Recall of the seed-based approach is much lower than that of the proposed methodology, indicating more false negatives detected by the seed-based approach. Based on the average F1 value, it can also be concluded that the proposed crack detection methodology yields better crack detection results than the seed-based approach.

4.1.4 *Limitations*

The limitation of the proposed methodology is that it relies on detecting elevation difference in local image regions to interpret the presence of cracks, therefore the performance might be deteriorated when detecting shallow cracks. Future research effort will be devoted to *i*) extracting cracking properties such as crack width from the detected crack contours in an accurate and efficient manner; and *ii*) exploring the possibility of fusing the intensity image and range image as complementary parts to improve crack detection performance.

Table 4-1. Summary of test results

	Category				Performance measures of the proposed methodology			Performance measures of the seed-based approach*		
Image index	Cracking pattern	Grooved pattern	Elevation variation	Noise level	Precision (%)	Recall (%)	F1 (%)	Precision (%)	Recall (%)	F1 (%)
1	Transverse cracking	YES	Medium	High	91.1	92.6	91.8	89.9	81.3	85.4
2	Punchout	NO	High	Medium	88.1	99.3	93.4	89.8	68.1	77.4
3	Corner break	NO	High	Extremely High	95.8	99.4	97.6	93.0	78.0	84.9
Average					91.7	97.1	94.3	90.9	75.8	82.6

*: Zhou, et al. [28]

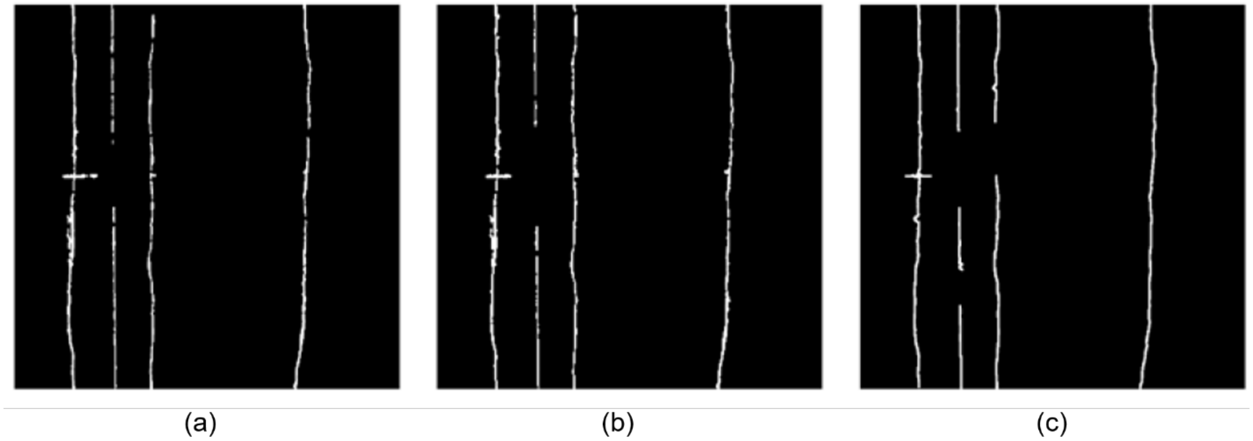


Figure 4-6. Binary crack maps of Surface 1: (a) ground truth by manual selection; (b) detected by the proposed methodology; and (c) detected by the seed-based approach.

4.2 Deep Learning-Based Crack Classification

Section 4.2.1 first briefly introduces the data generation process; and then, the experimental setup and the associated results and discussions are presented in section 4.2.2 and 4.2.3, respectively. Finally, the limitation of the proposed DCNN-based crack classification methodology is explained in section 4.2.4.

4.2.1 *Data Generation*

- Image acquisition and processing

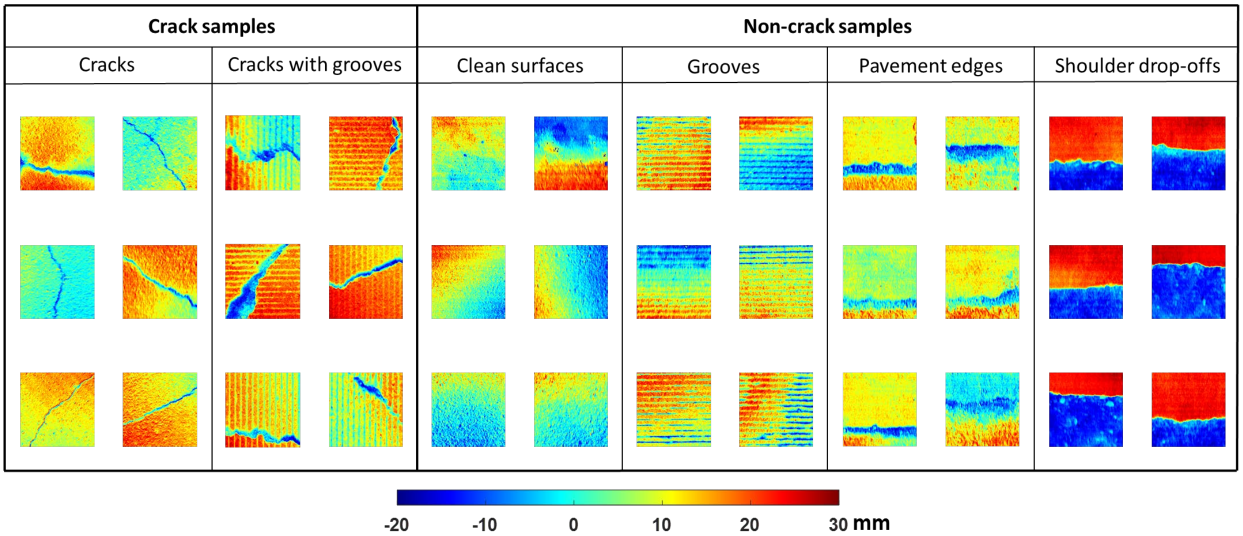


Figure 4-7. Samples of the collected range images.

The roadway image data used for analysis was collected by the laser imaging system (section 2.4) during an over one-year period through 2018 and 2019 on multiple concrete roads in Iowa. The sliding widow technique as described in section 2.3.2 is adopted to crop the roadway image data into image patches (dimension: 256×256×1). It is noted that this methodology directly works with raw range image data. No data pre-processing techniques (e.g., noise removal and crack enhancement) have been applied to the collected image data. Figure 4-7 demonstrates some data samples that are collected for this study. As can be observed in this figure, the image samples are highly diverse, which include cracks (in clean surfaces and grooved surfaces) and non-crack samples (clean surfaces, grooves, pavement edges, and shoulder drop-offs). It is worth noting that, most of these image samples suffer from varying levels of surface variations as well as scanning noises, as can be interpreted by the color variations.

- Ground truth generation

Each acquired image sample is classified by trained personnel to generate a ground truth label (i.e., “crack” vs. “non-crack”).

- Data augmentation

Data augmentation techniques (section 2.3.3) including random translation and rotation are used to effectively increase the number of crack samples. Each crack image is augmented through random translations along upward, downward, left, and right directions for a random distance within 100 pixels; additionally, augmented images are created by counterclockwise rotating the original image by 45°, 90°, and 135°.

- Dataset configuration

In total, 25785 crack samples (including the augmented images) and 26623 non-crack samples are obtained for training and testing purposes. The collected range images are separated into three datasets, which are the training dataset, validation dataset, and test dataset 1, respectively. Images in the training dataset are used to fit the DCNN model. During training, the fitted model routinely (e.g., every half an epoch) makes predictions on the validation dataset, which provides an evaluation on the goodness of the model fit and indicates if overfitting occurs or not. The test dataset is used after the training is completed to evaluate the performance among different networks. Shahin, et al. [142] investigated the issue of data division and its impact on network model performance. According to their case study, the optimal performance was obtained when 20% of the data were used for validation, 70% of the remaining data for training and 30% for testing. The data split ratio adopted in this study is 60% for training, 20% for validation, and 20% for testing, similar as obtained in the optimal case in [142]. Meanwhile, the ratio between crack and non-crack samples is around 1:1 to maintain a balanced class in the dataset. The number of images in each dataset is tabulated in Table 4-2.

Table 4-2. Number of image samples in each dataset.

	Training dataset	Validation dataset	Test dataset 1	Test dataset 2
Crack	15785	5000	5000	5000
Non-crack	16623	5000	5000	5000
Total	32408	10000	10000	10000

Additionally, another range image dataset, denoted as “test dataset 2”, is collected by the laser imaging system (section 2.4) for testing and demonstration purposes. It consists of 5000 cracks and 5000 non-crack samples. This laser-scanned range image dataset is very different than the training dataset and test dataset 1 by the following facts: *i*) locations: the test dataset 2 is acquired in Alabama which is different than the location where the training, validation, and test dataset 1 are obtained; *ii*) roadway types: the test dataset 2 is acquired on both asphalt and concrete roadways, which contains crack patterns that are different than test dataset 1, such as the alligator cracks illustrated in Figure 4-8. The purpose of introducing such an additional dataset for testing is to demonstrate the conclusions regarding the proposed DCNNs and associated hyperparameter selection for laser-scanned range images are consistent among diverse range image data, which provides the necessary real-world complexity to better explain the performance of the proposed DCNNs. The test dataset 2 [130] is made publicly available to benefit the community.

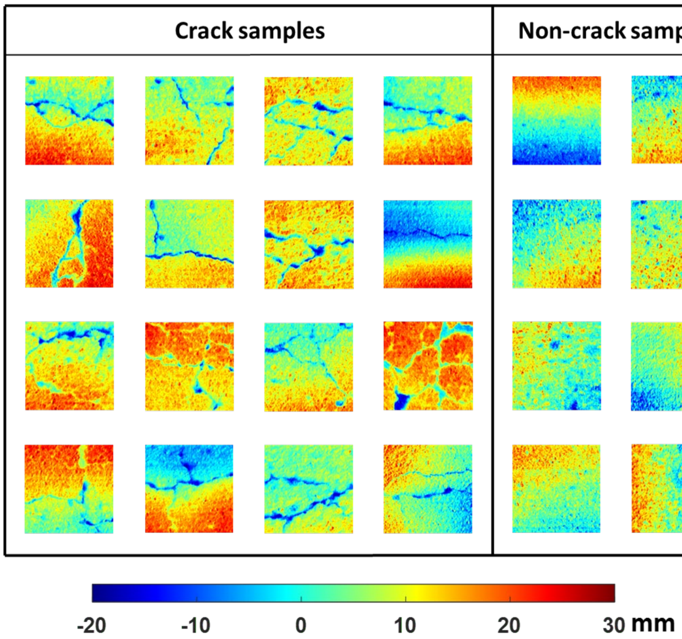


Figure 4-8. Samples of the test dataset 2.

4.2.2 *Experimental Setup*

- Computing hardware and software

All the experiments are performed on the same computer with the following specifications: CPU: Intel i7-8750H; GPU: Nvidia GTX 1060 with 6GB RAM. The proposed methodology is implemented in MATLAB environment [143]; its deep learning toolbox has also been used.

- Parameter initialization

The learnable parameters in the proposed DCNNs are initialized based on the specifications in section 2.3.4.

4.2.3 *Results and Discussions*

Two experiments are designed to determine the joint hyperparameter specification for an optimal DCNN architecture and the associated training scheme for roadway crack classification using laser-scanned range images. The first experiment, Case I, intends to explore the optimal network architecture by varying the hyperparameters related with network structure; Case II is focused on determining the optimal values for the hyperparameters related with training. In addition, two more experiments, Case III and IV, are implemented to further validate and demonstrate the network performance. Case III is a comparative study on the classification performance between the proposed architecture and four benchmark architectures. Case IV demonstrates the crack classification performance of the proposed architecture with a new set of range image data from actual roadway surveys.

- Case I: impacts from different kernel sizes, network depths and widths

Case I is designed to evaluate the impacts by changing the hyperparameters related with network structure. The first hyperparameter considered in Case I is the network depth. In a neural network, lower convolutional layers extract low-level features such as edges and color, while deeper layers extract shapes and texture. Increasing the number of convolutional/fully connected layers allows the network to adapt to high-level features but can meanwhile increase the computational cost. Pauly, et al. [129] showed that the network with a deeper architecture had higher classification performance on crack detection. The second hyperparameter is the kernel size in each convolutional layer. Convolutions with large filter kernels are beneficial in terms of their expressiveness and ability to extract features at a large scale, but the computation is disproportionately expensive [144]. In literature, it is reported [30] that smaller kernel sizes have better performance on crack detection than larger kernels. The third hyperparameters to be evaluated is the network width, which refers to the number of kernels in each layer. Enlarging the width of each convolutional layer can enhance the model capacity to represent the hierarchical features, but it can affect the training process by introducing more parameters which require stronger regularization. In a defects detection study, Faghih-Roohi, et al. [128] showed that a wider (i.e., a larger number of kernels) architecture had better performance but required a longer training time. Nevertheless, a comprehensive discussion on the optimal joint specification on these hyperparameters for deep learning-based roadway crack classification using laser-scanned range images is still missing in the literature.

The network configurations considered in Case I are tabulated in Table 4-3. As shown in this table, three hyperparameters including the network depth, kernel size, and network width are varied among different architecture candidates. To evaluate the effects of the network depths, architectures with different depths from five to eight weight layers (i.e., convolutional layers and fully connected layers) are created. Regarding the variations of kernel sizes, three patterns are considered: i) the sizes of the filter kernels gradually increase as the network goes deeper (i.e., an ascending trend); that is, the network intends to extract larger objects of interest in deeper layers; ii) the kernel sizes gradually decrease along the convolutional layers (i.e., a descending trend); and iii) the kernel sizes stay the same among all convolutional layers (i.e., a constant trend). Similarly, three patterns are considered for the network widths; that is, the number of kernels increases/decreases/stays the same from lower convolutional layers to deeper ones. The purpose of varying the network width is to investigate the effects of putting emphasis on extracting different levels of features.

Accordingly, the total number of architectures studied in Case I is 36. The symbols "↘", "↗", and "→" in Table 4-3 denote "descending", "ascending" and "constant" trends, respectively; the operator $\mathrm{Conv}(a, b)$ denotes that in the convolutional layer, the kernel size is $a \times a$ and the network width (i.e., number of kernels) is b; note that each $\mathrm{Conv}()$ operation consists of a convolutional layer followed by a batch normalization layer and a LReLU layer; the stride for each convolutional layer is 2, except that the last convolutional layer in the architecture uses 1×1 kernels with a stride of 1; different padding sizes are assigned for each convolutional layer to match the specific dimension of the feature map, as shown in Table 4-3; $\mathrm{FC}(x)$ denotes that the number of neuron output from the fully connected layer is x; $\mathrm{Softmax}(y)$ denotes that the number of classes is y. An example of the architecture is illustrated in Figure 3-10.

To isolate the influence, the hyperparameters associated with training are configured with the same setting among different candidates in Case I. The initial learning rate is selected as 0.01, and the learning rate drop factor is set as 0.8. In LReLU layers (section 2.3.1), the gradient for negative neuron input $\alpha = 0.01$ is selected, because this value is widely used in existing studies [38,90,145-147]. The dropout factor is fixed as 50%. According to [92], the probability to drop out each neuron is usually set as 50%, which is close to optimal for a wide range of networks and tasks. The mini-batch size is 60; that is, during each iteration, a subset (i.e., 60 images) of the training dataset is used to update the learnable parameters. One epoch is a full pass through the entire training dataset using mini-batches. As indicated by Table 4-2, the number of images in the training dataset is 32408; thus, the total number of iterations in each epoch is 540. The number of epochs is set as 30, which allows a sufficient amount of iterations for the training to converge. The weight decay and momentum are set as 0.0003 and 0.9, respectively, because these values are widely used in existing literature [30,38,39,42,129]. Discussion in Case II will show that such hyperparameter values for the mini-batch sizes, learning rates, dropout factors, and LReLU factors can yield the highest classification performance.

Table 4-3. Case I: network configuration.

Architecture	Hyperparameters			Layer type									
	Network depth	Kernel size	Network width	Conv	Conv	Conv	Conv	Conv	Conv	Dropout	Conv	FC	Softmax
				Input dimension									
				256 ×256	128×128	64×64	32×32	16×16	8×8				
1	5	↘	↗	(11,16)	(7,32)	(3,48)	-	-	-	50%	(1,2)	2	2
2	5	↘	↘	(11,48)	(7,32)	(3,16)	-	-	-	50%	(1,2)	2	2
3	5	↘	→	(11,48)	(7,48)	(3,48)	-	-	-	50%	(1,2)	2	2
4	5	↗	↗	(3,16)	(7,32)	(11,48)	-	-	-	50%	(1,2)	2	2
5	5	↗	↘	(3,48)	(7,32)	(11,16)	-	-	-	50%	(1,2)	2	2
6	5	↗	→	(3,48)	(7,48)	(11,48)	-	-	-	50%	(1,2)	2	2
7	5	→	↗	(7,16)	(7,32)	(7,48)	-	-	-	50%	(1,2)	2	2
8	5	→	↘	(7,48)	(7,32)	(7,16)	-	-	-	50%	(1,2)	2	2
9	5	→	→	(7,48)	(7,48)	(7,48)	-	-	-	50%	(1,2)	2	2
10	6	↘	↗	(15,16)	(11,32)	(7,48)	(3,64)	-	-	50%	(1,2)	2	2
11	6	↘	↘	(15,64)	(11,48)	(7,32)	(3,16)	-	-	50%	(1,2)	2	2
12	6	↘	→	(15,48)	(11,48)	(7,48)	(3,48)	-	-	50%	(1,2)	2	2
13	6	↗	↗	(3,16)	(7,32)	(11,48)	(15,64)	-	-	50%	(1,2)	2	2
14	6	↗	↘	(3,64)	(7,48)	(11,32)	(15,16)	-	-	50%	(1,2)	2	2
15	6	↗	→	(3,48)	(7,48)	(11,48)	(15,48)	-	-	50%	(1,2)	2	2
16	6	→	↗	(7,16)	(7,32)	(7,48)	(7,64)	-	-	50%	(1,2)	2	2
17	6	→	↘	(7,64)	(7,48)	(7,32)	(7,16)	-	-	50%	(1,2)	2	2
18	6	→	→	(7,48)	(7,48)	(7,48)	(7,48)	-	-	50%	(1,2)	2	2
19	7	↘	↗	(19,16)	(15,32)	(11,48)	(7,64)	(3,80)	-	50%	(1,2)	2	2
20	7	↘	↘	(19,80)	(15,64)	(11,48)	(7,32)	(3,16)	-	50%	(1,2)	2	2
21	7	↘	→	(19,48)	(15,48)	(11,48)	(7,48)	(3,48)	-	50%	(1,2)	2	2
22	7	↗	↗	(3,16)	(7,32)	(11,48)	(15,64)	(19,80)	-	50%	(1,2)	2	2
23	7	↗	↘	(3,80)	(7,64)	(11,48)	(15,32)	(19,16)	-	50%	(1,2)	2	2
24	7	↗	→	(3,48)	(7,48)	(11,48)	(15,48)	(19,48)	-	50%	(1,2)	2	2
25	7	→	↗	(7,16)	(7,32)	(7,48)	(7,64)	(7,80)	-	50%	(1,2)	2	2
26	7	→	↘	(7,80)	(7,64)	(7,48)	(7,32)	(7,16)	-	50%	(1,2)	2	2
27	7	→	→	(7,48)	(7,48)	(7,48)	(7,48)	(7,48)	-	50%	(1,2)	2	2
28	8	↘	↗	(23,16)	(19,32)	(15,48)	(11,64)	(7,80)	(3,96)	50%	(1,2)	2	2
29	8	↘	↘	(23,96)	(19,80)	(15,64)	(11,48)	(7,32)	(3,16)	50%	(1,2)	2	2
30	8	↘	→	(23,48)	(19,48)	(15,48)	(11,48)	(7,48)	(3,48)	50%	(1,2)	2	2
31	8	↗	↗	(3,16)	(7,32)	(11,48)	(15,64)	(19,80)	(23,96)	50%	(1,2)	2	2
32	8	↗	↘	(3,96)	(7,80)	(11,64)	(15,48)	(19,32)	(23,16)	50%	(1,2)	2	2
33	8	↗	→	(3,48)	(7,48)	(11,48)	(15,48)	(19,48)	(23,48)	50%	(1,2)	2	2
34	8	→	↗	(7,16)	(7,32)	(7,48)	(7,64)	(7,80)	(7,96)	50%	(1,2)	2	2
35	8	→	↘	(7,96)	(7,80)	(7,64)	(7,48)	(7,32)	(7,16)	50%	(1,2)	2	2
36	8	→	→	(7,48)	(7,48)	(7,48)	(7,48)	(7,48)	(7,48)	50%	(1,2)	2	2

Table 4-4. Case I: performance metrics.

Architecture	Training dataset	Validation dataset (at the end of training)			Test dataset 1				Test dataset 2				Number of parameters
	Time (min)	Precision (%)	Recall (%)	F1 (%)	Precision (%)	Recall (%)	F1 (%)	Time (sec)	Precision (%)	Recall (%)	F1 (%)	Time (sec)	
1	44	98.2	96.6	97.4	98.0	98.2	98.1	6.8	93.6	98.9	96.2	7.7	45332
2	71	97.4	97.6	97.5	97.1	98.5	97.8	10.8	91.6	99.4	95.3	11.6	90100
3	86	96.8	97.7	97.3	96.6	98.8	97.7	13.1	91.5	99.5	95.3	13.9	144068
4	61	99.2	96.6	97.9	99.2	98.3	98.7	9.1	94.1	97.9	96.0	8.4	215572
5	84	98.5	97.8	98.2	98.0	99.1	98.5	11.9	84.8	98.8	91.2	12.2	142068
6	104	97.8	97.7	97.7	97.8	98.2	98.0	16.4	88.8	99.1	93.6	15.6	396740
7	51	99.7	97.6	98.6	99.5	98.7	99.1	8.1	97.1	99.3	98.2	7.5	105620
8	71	98.7	98.5	98.6	98.7	99.0	98.9	11.1	95.7	99.5	97.6	11.6	107124
9	90	99.2	97.6	98.4	98.9	98.6	98.8	13.2	95.3	99.3	97.2	14.4	232772
10	91	99.3	98.8	99.0	99.3	99.1	99.2	11.2	92.2	99.2	95.6	10.8	170100
11	197	99.3	98.7	99.0	98.6	99.3	98.9	28.9	96.8	99.1	97.9	30.0	467524
12	159	99.2	98.7	98.9	98.7	99.2	99.0	22.9	92.5	99.3	95.7	25.1	424916
13	84	99.6	99.2	99.4	99.6	99.2	99.4	13.8	98.7	99.2	98.9	13.6	903924
14	123	99.4	99.2	99.3	99.6	99.3	99.5	19.7	96.7	99.4	98.1	18.7	453700
15	117	99.7	98.8	99.3	99.6	99.2	99.4	18.6	97.2	99.2	98.2	20.3	912212
16	71	99.7	99.4	99.5	99.5	99.4	99.5	7.9	94.9	99.6	97.2	7.8	253300
17	108	99.6	99.0	99.3	99.4	99.2	99.3	17.6	97.1	99.8	98.4	16.4	255556
18	93	99.7	98.7	99.2	99.5	99.0	99.3	14.6	98.8	99.5	99.1	14.4	342740
19	151	99.7	99.5	99.6	99.5	99.4	99.5	22.8	99.3	99.4	99.3	19.4	504580
20	550	99.3	98.9	99.1	99.4	99.2	99.3	111.7	98.4	98.8	98.6	107.3	1633476
21	327	99.5	98.9	99.2	99.2	99.3	99.2	61.6	98.1	99.3	98.7	65.9	949220
22	87	99.5	99.4	99.4	99.7	99.4	99.5	13.7	97.1	99.4	98.2	15.3	2751748
23	167	99.4	99.4	99.4	99.5	99.2	99.4	30.0	96.6	99.3	97.9	27.1	1154756
24	126	99.5	98.6	99.1	99.5	99.0	99.2	21.4	95.9	97.4	96.6	22.2	1743332
25	71	99.7	99.5	99.6	99.7	99.6	99.6	7.9	99.5	99.5	99.5	7.8	503684
26	144	99.8	99.5	99.6	99.5	99.4	99.5	26.3	99.1	99.6	99.4	21.2	506692
27	95	99.8	99.3	99.5	99.6	99.3	99.5	16.1	98.5	99.3	98.9	14.7	455012
28	249	99.6	98.0	98.8	99.5	99.1	99.3	35.2	98.7	98.9	98.8	31.3	1231876
29	1355	99.1	98.2	98.6	99.0	99.1	99.0	224.3	98.8	98.9	98.9	232.8	4427956
30	539	98.8	98.2	98.5	99.2	99.1	99.1	102.8	99.1	97.6	98.3	103.8	1788980
31	116	99.6	98.4	99.0	99.7	99.1	99.4	18.4	93.6	98.7	96.1	15.9	6814596
32	262	99.8	97.9	98.9	99.8	98.9	99.3	45.9	94.3	96.8	95.5	41.0	2514356
33	132	98.7	97.6	98.1	99.2	98.5	98.8	21.7	93.4	98.4	95.9	21.7	2962100
34	73	99.7	99.2	99.4	99.5	99.5	99.5	8.0	98.6	99.4	99.0	8.0	880132
35	219	99.6	99.5	99.5	99.4	99.4	99.4	38.7	99.3	98.8	99.1	33.0	883892
36	96	99.7	99.2	99.4	99.5	99.4	99.5	16.1	99.2	99.4	99.3	14.7	567860

Table 4-5. Case I: the effects of changing the kernel sizes.

Architecture	Kernel size	Training dataset	Validation dataset (at the end of training)			Testing dataset 1				Testing dataset 2				Number of parameters
		Time (min)	Precision (%)	Recall (%)	F1 (%)	Precision (%)	Recall (%)	F1 (%)	Time (sec)	Precision (%)	Recall (%)	F1 (%)	Time (sec)	
25	7×7	71	99.7	99.5	99.6	99.7	99.6	99.6	7.9	99.5	99.5	99.5	7.8	503684
25-A	3×3	35	99.4	97.8	98.6	99.0	98.4	98.7	5.8	91.7	98.4	95.0	5.9	93444
25-B	11×11	86	99.8	99.0	99.4	99.7	99.1	99.4	11.3	98.9	99.4	99.2	11.3	1242116

Table 4-6. Case II: performance metrics.

Architecture	Mini-batch size	Dropout factor	LReLU factor	Training dataset	Validation dataset (at the end of training)			Testing dataset 1				Testing dataset 2			
				Time (min)	Precision (%)	Recall (%)	F1 (%)	Precision (%)	Recall (%)	F1 (%)	Time (sec)	Precision (%)	Recall (%)	F1 (%)	Time (sec)
25	60	0.5	0.01	71	99.7	99.5	99.6	99.7	99.6	99.6	7.9	99.5	99.5	99.5	7.8
25	10	0.5	0.01	143	99.9	99.5	99.7	99.8	99.5	99.6	7.7	99.3	99.3	99.3	7.8
25	20	0.5	0.01	103	99.8	99.6	99.7	99.5	99.6	99.5	7.8	99.5	99.6	99.5	7.8
25	30	0.5	0.01	85	99.8	99.6	99.7	99.6	99.6	99.6	7.8	99.1	99.6	99.4	7.8
25	40	0.5	0.01	74	99.8	99.5	99.6	99.5	99.6	99.5	7.7	99.2	99.7	99.5	7.9
25	50	0.5	0.01	74	99.8	99.6	99.7	99.6	99.5	99.5	7.8	99.2	99.4	99.3	7.8
25	70	0.5	0.01	70	99.8	99.5	99.7	99.7	99.2	99.5	7.7	99.3	99.3	99.3	7.8
25	80	0.5	0.01	70	99.7	99.5	99.6	99.7	99.4	99.5	7.8	98.5	99.8	99.1	7.8
25	90	0.5	0.01	71	99.6	99.4	99.5	99.6	99.5	99.6	7.8	97.6	99.7	98.7	7.8
25	100	0.5	0.01	73	99.8	99.4	99.6	99.6	99.3	99.4	7.7	98.6	99.6	99.1	7.9
25-C	60	N/A	0.01	72	99.6	98.9	99.2	99.6	99.3	99.5	8.0	98.6	99.2	98.9	7.8
25-D	60	0.1	0.01	71	99.6	99.3	99.5	99.6	99.4	99.5	7.9	98.6	99.6	99.1	7.8
25-E	60	0.3	0.01	71	99.8	99.3	99.5	99.6	99.3	99.4	7.8	99.5	99.4	99.4	7.8
25-F	60	0.7	0.01	71	99.9	99.3	99.6	99.6	99.3	99.4	7.9	99.1	99.7	99.4	7.8
25-G	60	0.9	0.01	71	99.4	98.6	99.0	99.4	99.1	99.3	8.2	98.7	99.4	99.1	7.8
25-H	60	0.5	0.001	71	99.8	99.4	99.6	99.5	99.3	99.4	7.8	98.7	99.7	99.2	7.8
25-I	60	0.5	0.005	71	99.7	99.4	99.5	99.5	99.3	99.4	7.8	99.1	99.4	99.3	7.8
25-J	60	0.5	0.05	71	99.8	99.6	99.7	99.6	99.4	99.5	7.8	99.2	99.6	99.4	7.8
25-K	60	0.5	0.1	71	99.8	99.7	99.7	99.7	99.3	99.5	7.9	98.5	99.8	99.1	7.8
25-L	60	0.5	0.5	71	97.3	94.1	95.6	98.4	96.4	97.4	7.9	85.3	98.4	91.4	7.8

Table 4-7. Case III: performance metrics.

Architecture	Training dataset	Validation dataset (at the end of training)			Testing dataset 1				Testing dataset 2				Number of parameters ($\times 10^6$)
	Time (min)	Precision (%)	Recall (%)	F1 (%)	Precision (%)	Recall (%)	F1 (%)	Time (sec)	Precision (%)	Recall (%)	F1 (%)	Time (sec)	
25	71	99.7	99.5	99.6	99.7	99.6	99.6	7.9	99.5	99.5	99.5	7.8	0.5
Resnet18	217	99.8	99.8	99.8	99.7	99.9	99.8	47.0	100.0	99.3	99.6	47.2	11.7
Resnet34	379	99.8	99.6	99.7	99.8	99.7	99.7	63.1	99.9	99.4	99.6	64.0	21.8
Resnet50	668	99.8	99.8	99.8	99.7	99.7	99.7	98.3	99.9	99.6	99.8	99.6	25.6
Resnet101	1340	99.7	99.8	99.8	99.8	99.9	99.8	145.5	99.7	99.8	99.8	146.2	44.5

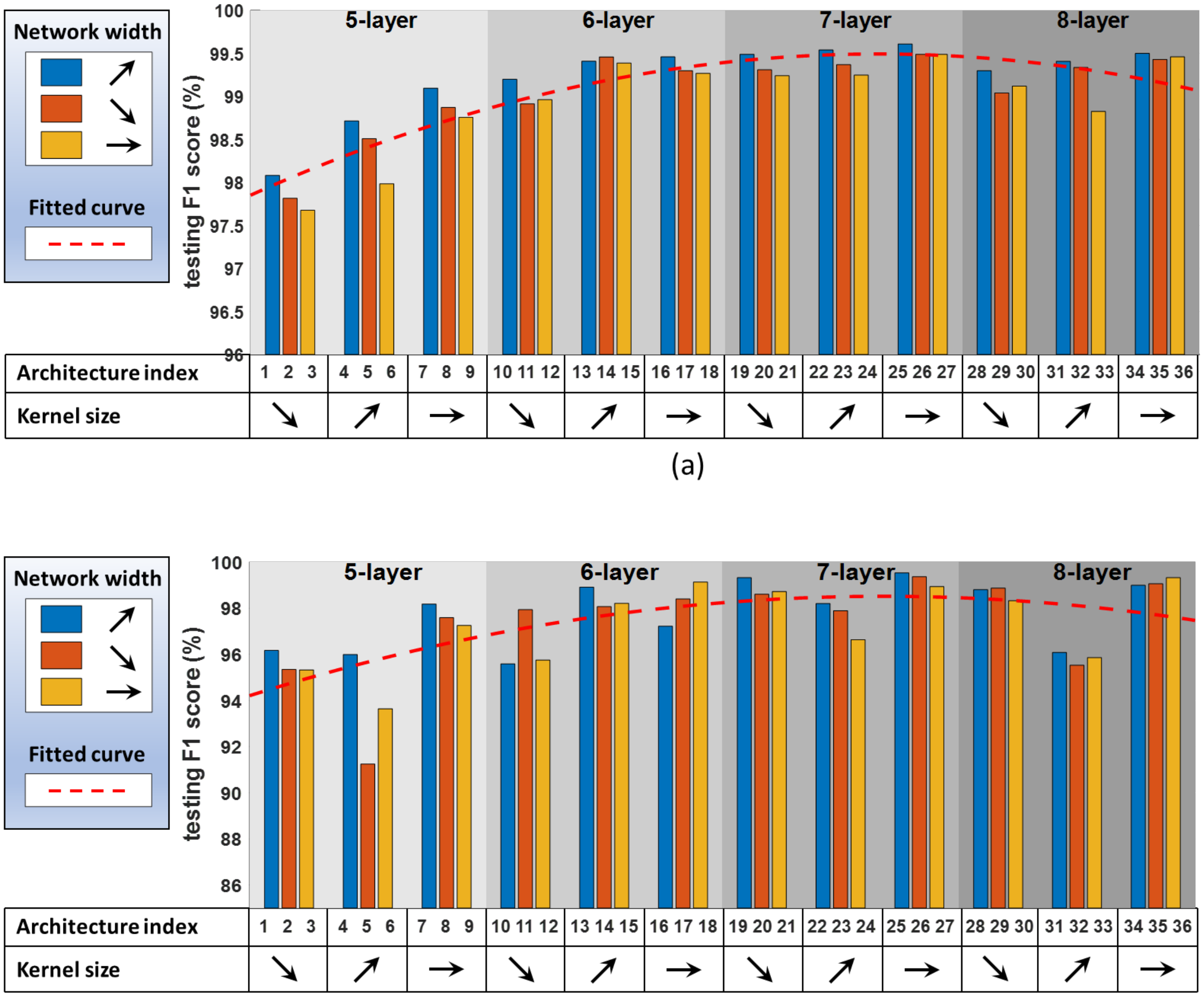
The performance metrics used for comparison are the Precision, Recall, F1 score (section 2.5.1), training time, testing time, and number of parameters. Detailed statistics of these metrics are tabulated in Table 4-4. To highlight, the F1 scores on the test datasets 1 and 2 are illustrated in Figure 4-9 (a) and (b), respectively. Note that the horizontal axis of this figure is the architecture index, as defined in Table 4-3. These indexes are further grouped by the patterns of variations in the kernel sizes, referred to as the descending ("↘")/ascending ("↗")/constant ("→") trends, respectively. In this figure, blue/orange/yellow colors represent the architectures with ascending ("↗")/descending ("↘")/constant ("→") network widths, respectively. Meanwhile, a curve is fit to the calculated F1 values, as indicated by the red dashed line in both plots.

Several observations regarding the impact from different architecture layouts through hyperparameter selection can be made from Figure 4-9 (a) and (b) by comparing the F1 scores on the test datasets:

i) By investigating the networks with different depths, it can be observed from the F1 statistics on both test datasets that better classification performance can be achieved as the network depth increases from 5 layers until reaching 7 weight layers. For example, the average F1 score increases from 98.4% (5-layer architectures) to 99.4% (7-layer architectures) for the test dataset 1, and from 95.6% to 98.6% for the test dataset 2, respectively. A similar trend can be observed from the fitted curves as well, where the peak occurs at 7-layer architectures for both test datasets;

ii) On the comparison between the architectures with the same depths but different trends of kernel sizes, the ones using constant kernel sizes yield the best performance, which achieve higher F1 scores by an average margin of 0.4% for the test dataset 1 and 1.7% for the test dataset 2 than the ones with descending/ascending kernel sizes;

iii) The architectures with ascending network widths (blue color) generally outperform the ones with descending/constant widths on the F1 scores by an average of 0.3% for the test dataset 1 and 0.5% for the test dataset 2, which suggests it is beneficial to put more emphasis on extracting the high-level features.

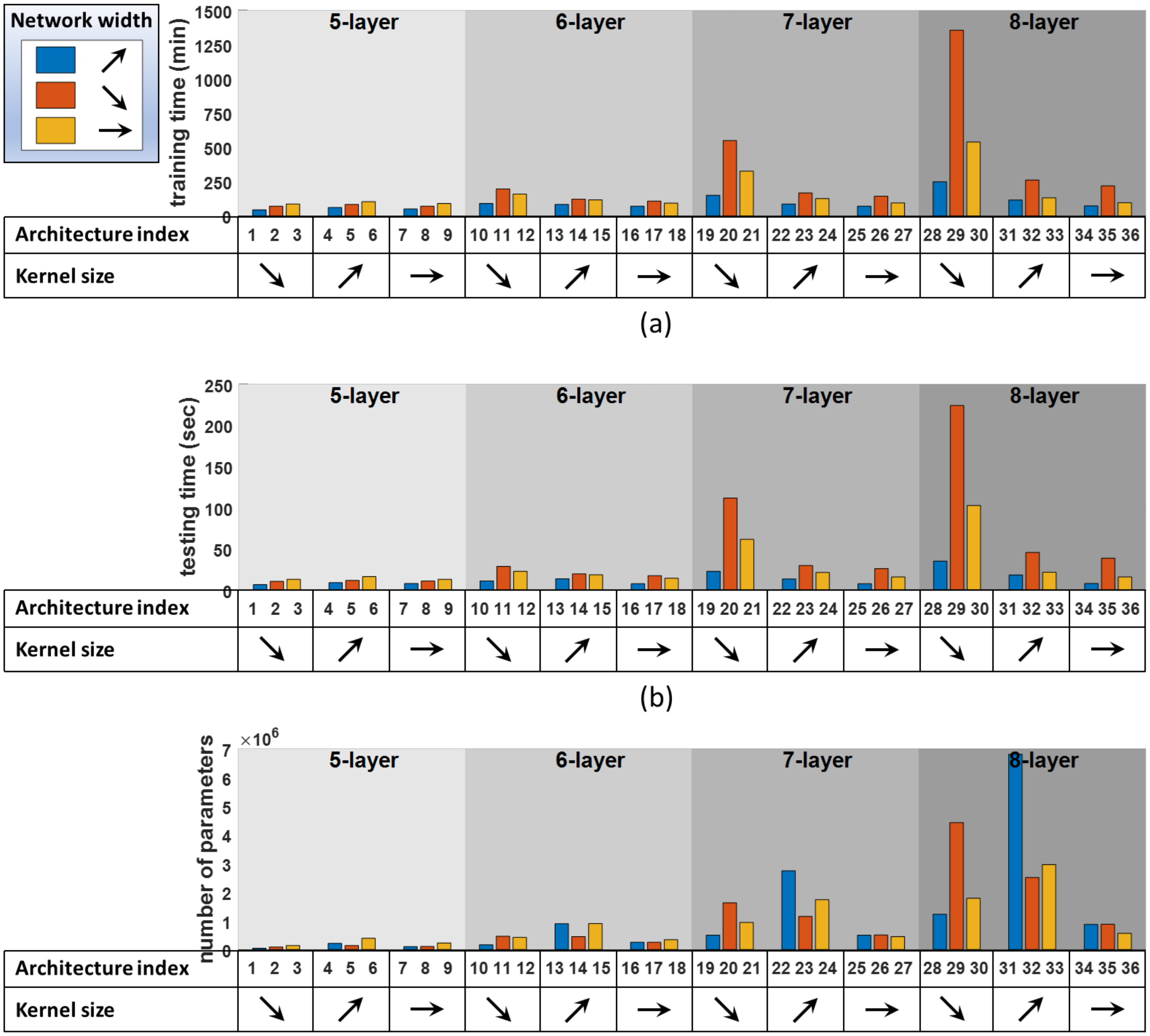


(a)

(b)

Figure 4-9. Case I: performance metrics: (a) F1 score on the test dataset 1; and (b) F1 score on the test dataset 2.

Judging from the highest F1 scores on the test datasets 1 and 2, the No. 25 candidate, which is a 7-layer architecture with 7×7 convolutional kernels and ascending network widths, is determined as the optimal architecture for the laser-scanned range image datasets. The reason the No. 25 architecture yields the highest classification performance among the 36 candidates may be as follows: among a family of DCNN classifiers proposed in this study (see Table 4-3), the No. 25 architecture can best describe and reflect the real-world complexities of the 3D laser-scanned roadway range image data, which are collected from the fields under practical conditions.



(a)

(b)

(c)

Figure 4-10. Case I: performance metrics: (a) training time; (b) testing time on the test dataset 1; and (c) number of parameters.

The other performance metrics including the training time, testing time, and number of parameters are further evaluated to help determine the optimal architecture. The training time, testing time, and number of parameters for each architecture are illustrated in Figure 4-10 (a), (b), and (c), respectively. Only the testing time on the test dataset 1 is illustrated for concision, because the statistics of testing time on the test datasets 1 and 2 are very close. The color notation and horizontal axis labels are the same as defined in Figure 4-9. From Figure 4-10 (a) and (c), it can be seen that the training time increases as the network architecture becomes deeper, introducing more parameters to participate in the training. For example, by gradually increasing the network depths through No. 5, 14, 23, and 32 architectures, the training and testing time increase from 84 mins to 262 mins, and 11.9 secs to 45.9 secs, respectively. Variations in the network widths also lead to a significant impact on the network efficiency. As an example, for the No. 22, 23, and 24 architectures, which have the same depths and kernel sizes but ascending/descending/constant network widths, the corresponding training and testing time are 87, 167, and 126 mins, and 13.7, 30.0, and 21.4 secs, respectively. In general, the architectures with ascending network widths (blue color) lead to the most efficient training [Figure 4-10 (a)] and testing [Figure 4-10 (b)] performance (i.e., an average of 96 mins training time and 13.6 secs testing time); on the contrary, the architectures with descending network widths result in the worst efficiency (i.e., an average of 279 mins training time and 48.1 secs testing time). This phenomenon implies that employing a larger number of kernels in the lower layers will cause more computational efforts on the convolution operation, which drastically deteriorates the efficiency of the network.

Based on the above observations, a series of conclusions can be summarized: *i*) the hyperparameters including the network depths, kernel sizes, and network widths can impact the accuracy and efficiency of the neural networks to a significant extent; *ii*) regarding the network depths, generally, a deeper architecture yields better performance but meanwhile requires more computational efforts; the optimal depth for the proposed crack classification DCNN for the collected laser-scanned roadway range image data is 7 weight layers; *iii*) using constant kernel sizes leads to the highest classification performance; *iv*) variations in the network widths have a large impact on the efficiency of the network; it is preferable to gradually increasing the number of kernels from lower layers to deeper layers (i.e., an ascending trend), which results in better performance on both the accuracy and efficiency than the other cases (i.e., descending and constant trends).

Furthermore, an additional study is performed to explore the effects of using medium-sized constant kernels vs. small/large kernels. Two variants of the No. 25 architecture are implemented. The only difference is that the kernel sizes are 7×7, 3×3, and 11×11 for the original No. 25 architecture and its variants No. 25-A, and 25-B, respectively. The corresponding performance metrics are tabulated in Table 4-5. It can be seen from this table that the architecture with medium-sized kernels (kernel size = 7×7) yields the highest F1 scores (1% improvement from No.25-A on the test dataset 1 and 4.5% on the test dataset 2), and meanwhile maintains an efficient training and testing speed.

- Case II: impacts from different mini-batch sizes, learning rates, dropout factors, and LReLU factors

With the optimal DCNN architecture (No. 25 in Table 4-3) selected in Case I, the effects of varying the mini-batch sizes, learning rates, dropout factors, and LReLU factors are investigated in Case II. Case II is comprised of four subcases as described below:

i) Different mini-batch sizes

The mini-batch size is the number of image data that are utilized in each iteration to update the parameters. It is common to train and test image datasets in small batches to improve the efficiency and generalization. Accordingly, the choice on the mini-batch size affects the convergence speed and computational efficiency. It has been observed in literature that when using a larger batch there is a degradation in the quality of the model, as measured by its ability to generalize [148]. In Case II, the effects of utilizing different mini-batch sizes for training are investigated.

Different mini-batch sizes varying from small to medium [Equation (4-1)] are adopted to train the optimal architecture (No. 25 in Table 4-3). The performance metrics are tabulated in Table 4-6 and illustrated in Figure 4-11 (a) and (b), respectively. As can be observed from Figure 4-11 (a), the training time are effectively reduced by up to 40% as the mini-batch size increases from 10 to 50, indicating an improvement of the network efficiency. Keeping increasing the mini-batch to 100 does not lead to a significant impact on the training efficiency. Meanwhile, from Figure 4-11 (b), which shows the F1 scores on the test datasets 1 and 2, it can be concluded that increasing the mini-batch sizes from 10 to 100 result in relatively small variations (≤0.5%) in the testing F1 score. By jointly considering the network performance in terms of efficiency and accuracy, the optimal value for the mini-batch size is selected as 60, which is used in Case I.

$$\text{Mini-batch sizes} = [\,10\,,20\,,30\,,40\,,50\,,60\,,70\,,80\,,90\,,100\,] \tag{4-1}$$

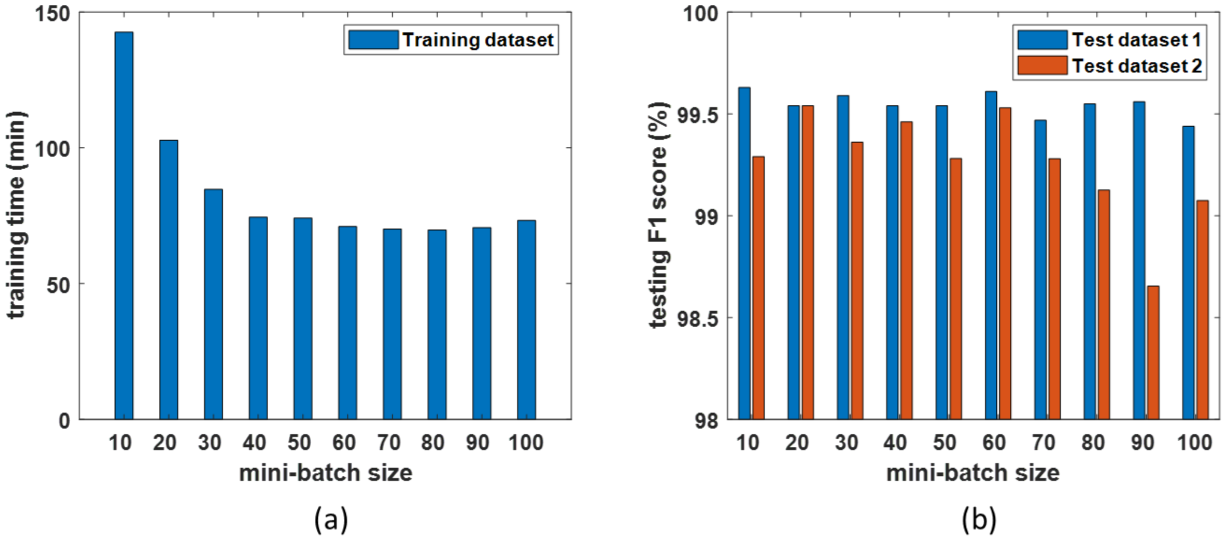


Figure 4-11. Case II: the effects of changing the mini-batch size: (a) training time; and (b) F1 score on the test datasets 1 and 2.

ii) Different initial learning rates and learning rate drop factors

The learning rate is considered as one of the most important hyperparameters that needs to be carefully tuned for the model [38]. Determination of the learning rate is a very critical and yet case-dependent procedure, as it affects the model ability to quickly adapt to the data through controlling the step size. A learning rate that is too large will result in a slow convergence or even instability, whereas a too small one can cause the parameters to stop responding to the error gradient. According to the recent observations [149,150], smaller learning rates lead to a sharper minima and poorer generalization. In the literature [149,151], large learning rates are recommended. In Case II, the proper learning rate for the proposed DCNN is determined.

This study utilizes a piecewise learning rate, which gradually decreases during training, to facilitate a quick convergence in the early stage of training and fine-tune the parameters in the late stage. The hyperparameters related with the learning rate are the initial learning rate and the learning rate drop factor. By using a piecewise learning rate strategy, the learning rate decays by multiplying a drop factor every a few iterations, for example, every two epochs. In this study, a grid search is employed to explore the optimal joint specification of the values related with the learning rates. A set of values for the initial learning rates and learning rate drop factors are specified as Equations (4-2a) and (4-2b), respectively. In total, 30 combinations of the initial learning rates and drop factors are utilized to train the optimal architecture. During training, the learning rate drops every two epochs. On evaluating the performance, the F1 scores evaluated on the test datasets 1 and 2 are employed as the metrics, which are illustrated in Figure 4-12 (a) and (b), respectively. Detailed statistics are not tabulated herein for concision. In each plot of Figure 4-12, the horizontal x axis is the initial learning rate value, and the horizontal y axis is the learning rate drop factor; the vertical z axis is the testing F1 score; in addition, a surface is fit to the data samples. It can be observed that as the learning rate drop factor decreases, the F1 scores on both test datasets will reduce correspondingly (by up to 4% in the test dataset 1 and 9% in the test dataset 2), especially when the initial learning rate is relatively small. With a large initial learning rate (e.g., between 0.005 and 0.05) and a high drop factor (e.g., 0.7 to 0.9), the resulted testing F1 scores will plateau to a high value (≥99%). The optimal point marked in Figure 4-12 corresponds to an initial learning rate of 0.01 and a drop factor of 0.8, which are used for the study in Case I.

$$\text{Initial learning rate } = [\ 0.001 \quad 0.005 \quad 0.01 \quad 0.05 \quad 0.1\ ] \tag{4-2a}$$

$$\text{Learning rate drop factor } = [\ 0.9 \quad 0.8 \quad 0.7 \quad 0.6 \quad 0.5 \quad 0.4\ ] \tag{4-2b}$$

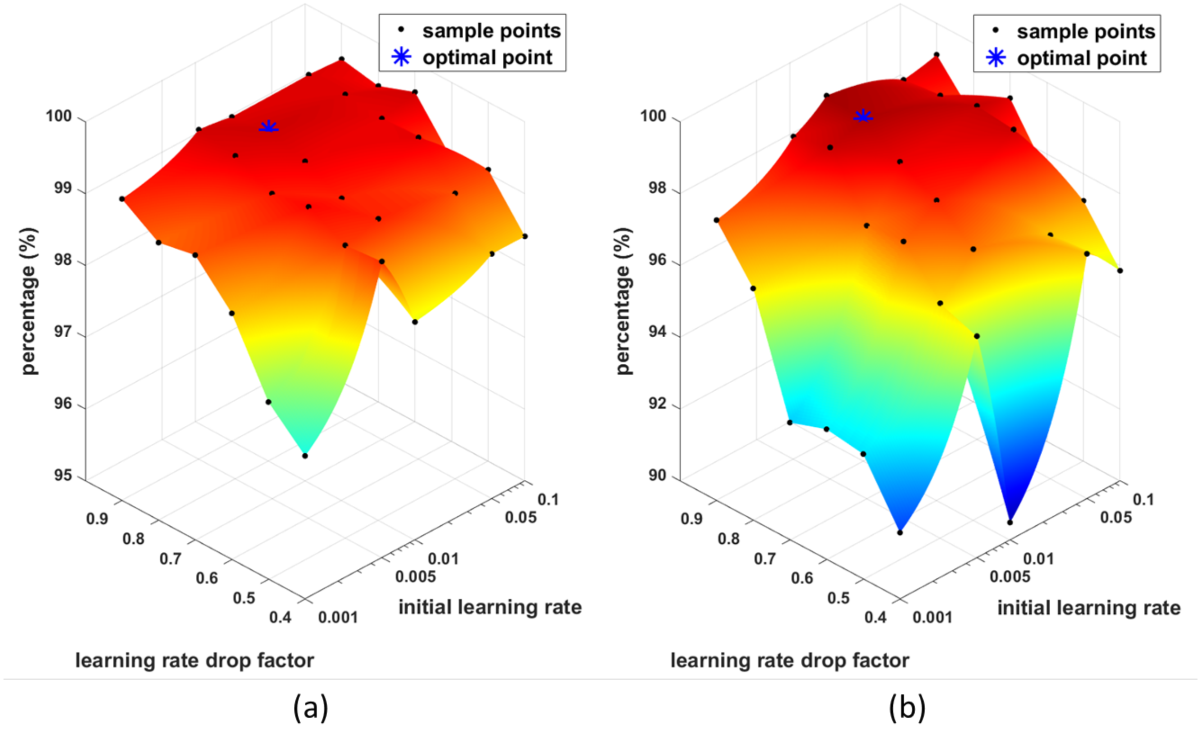


Figure 4-12. Case II: the effects of changing the learning rate: (a) F1 score on the test dataset 1; and (b) F1 score on the test dataset 2.

iii) Different dropout factors

In DCNN training, it may occur that a neural network is too attuned to the training data that it loses the ability to generalize to new data. Such a phenomenon is referred to as overfitting. Dropout [92] is an effective regularization technique to address this issue. During training, each neuron input to the dropout layer is randomly deactivated by a probability (i.e., dropout factor), such that the network is forced to adapt to different neurons, which improves generalization. In the literature [92], it is reported that a dropout factor of 50% is suitable for a wide variety of tasks. In Case II, five variants of the optimal architecture, denoted as No. 25-C through G, are developed, each having a different dropout factor in their dropout layer, as expressed in Equation (4-3). It is noted that a dropout factor of 0 indicates no dropout is applied to the neuron input; and, a dropout factor of 90% indicates that each neuron is very likely to be deactivated during training.

$$\text{Dropout factor } = [\,0 \quad 10\% \quad 30\% \quad 70\% \quad 90\%\,] \qquad \textbf{(4-3)}$$

The experimental results on the acquired laser-scanned range image datasets are tabulated in Table 4-6. From the table, varying the dropout factor does not impact the training or testing efficiency; also, judging from the validation and testing metrics, no overfitting is observed. The F1 scores on the test datasets 1 and 2 are illustrated in Figure 4-13 (a). It is observed that the dropout value corresponding to the highest F1 scores on the test datasets 1 and 2 is 50%, which is adopted by the optimal architecture in Case I. Moreover, as the dropout factor gradually increases from 0 to 50%

or gradually decreases from 90% to 50%, the network achieves better classification performance on the test datasets.

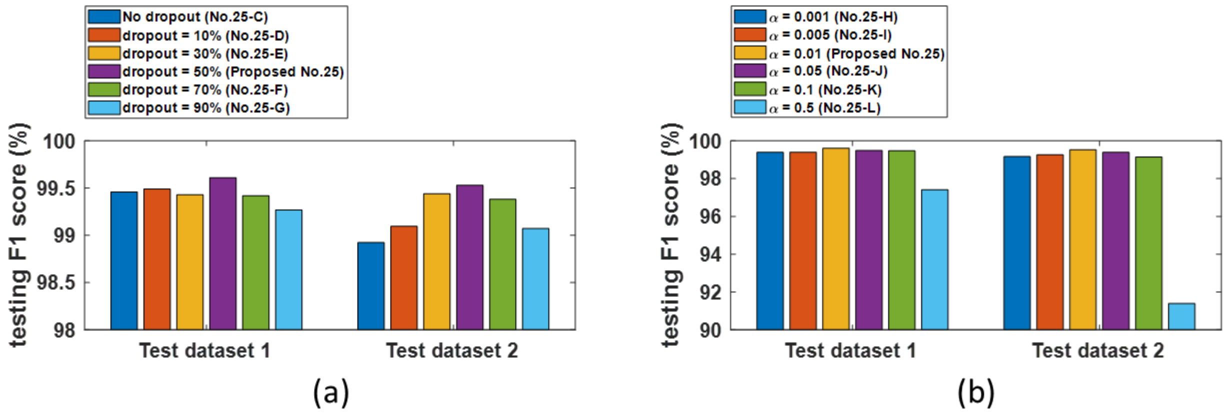


Figure 4-13. Case II: the effects of changing the dropout factor and LReLU factor: F1 score on the test datasets 1 and 2 upon changing (a) the dropout factor; and (b) the LReLU factor.

iv) Different LReLU factors

In literature, LReLU [90] is often utilized as the nonlinear activation function to add nonlinearity to a neural network. LReLU is a bi-linear function whose gradient for non-negative input is 1, and α for negative input. Usually, α is selected as a small positive value to avoid "dying neuron" problem during back-propagation. Five variants of the optimal architecture, denoted as No. 25-H through L, are developed to investigate the influence of changing the LReLU factor. Their LReLU factor values in each LReLU layer are expressed in Equation (4-4).

The corresponding performance metrics are displayed in Table 4-6. It is observed from this table that varying the LReLU factor does not lead to a notable impact on the training and testing efficiency on the collected laser-scanned range image datasets. Regarding the classification accuracy, the F1 scores on the test datasets 1 and 2 upon changing the LReLU factors are illustrated in Figure 4-13 (b). It can be seen when the LReLU factor is relatively small (≤ 0.1), the resulted F1 scores have a very marginal variation of less than 0.3%. However, when a large LReLU factor α = 0.5 is used, the classification accuracy deteriorates, as can be observed from the F1 scores on both test datasets. It is also noteworthy that the LReLU factor leading to the highest F1 scores on the test datasets is equal to 0.01, which is adopted in the optimal architecture in Case I.

$$\alpha = [\,0.001 \quad 0.005 \quad 0.05 \quad 0.1 \quad 0.5\,] \tag{4-4}$$

- Case III: proposed vs. benchmark architectures

This section illustrates the comparison result on the model performance between the optimal architecture (No.25) and four benchmark architectures as Resnet18, Resnet34, Resnet50, and Resnet101 [82], which represent different levels of model complexities. These four architectures are commonly used DCNNs for image classification and segmentation tasks. Figure 4-14 illustrates the comparison result, and the detailed statistics are presented in Table 4-7. Consistent results on the F1 score and testing time on two different test datasets are observed. It can be seen from Figure 4-14 (a) that the proposed architecture achieves similar levels (within a margin of 0.3%) of F1 scores on the test datasets 1 and 2 as the benchmark architectures. It is thus shown the proposed architecture is capable of achieving high accuracy on crack classification. Meanwhile, judging from Figure 4-14 (b-c), the proposed architecture yields the most efficient training and testing performance compared against the benchmarks; and, it is able to reflect the complexity of the range image data by using the least amount of learnable parameters, as shown in Figure 4-14 (d). For example, the No. 25 architecture only consumes around 5% of the training and testing time and 1% of the number of learnable parameters as consumed by Resnet101, but it can achieve the same level of classification accuracy (within a margin of 0.3% in the F1 scores on both test datasets).

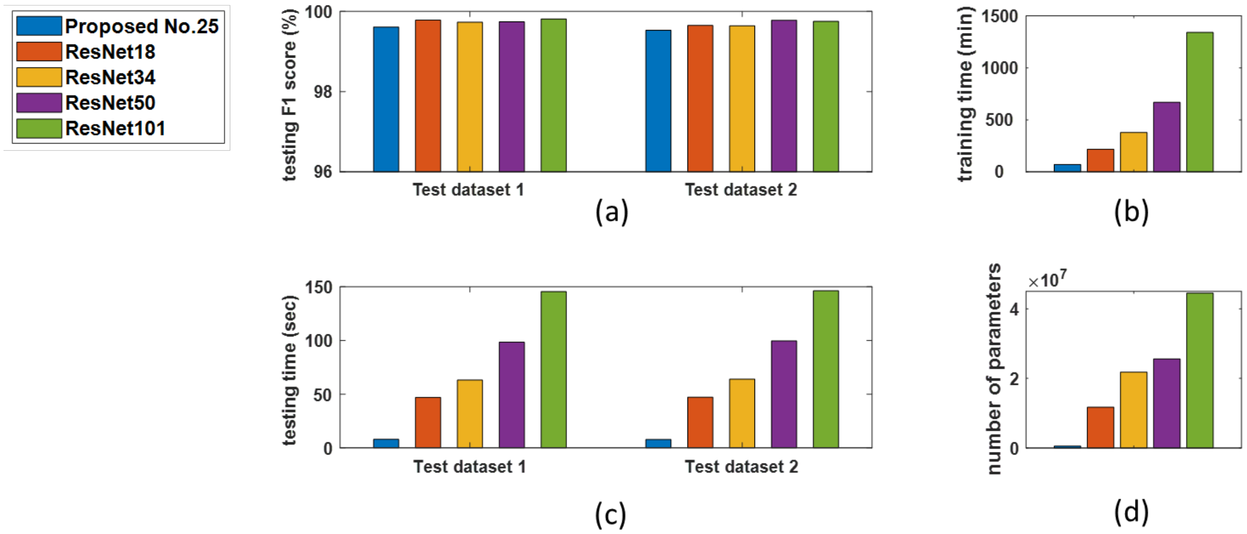


Figure 4-14. Case III: performance comparison: (a) F1 score on the test datasets 1 and 2; (b) training time; (c) testing time on the test datasets 1 and 2; and (d) number of parameters.

- Case IV: a crack classification example using the optimal architecture

To further demonstrate the classification performance of the proposed DCNN methodology, the trained optimal architecture is tested on another new dataset from actual roadway surveys. As illustrated in Figure 4-15, the range images in this dataset are collected from three roadway surfaces [Figure 4-15 (a), (b), and (c)] containing cracks and many non-crack patterns such as grooves and pavement edges. It is also worth noting that the range images suffer from drastic surface variations, indicated by the change of color. The image patches are collected by using the sliding window technique with a 50% overlap and predicted by the trained classifier. The predicted crack map for each surface is enclosed by a black line, and the corresponding testing F1 score is 99.6%, 99.8%, and 99.4%, respectively. It is shown that the proposed methodology can detect surface cracks with a high accuracy under the disturbance of many non-crack patterns such as grooves and pavement edges.

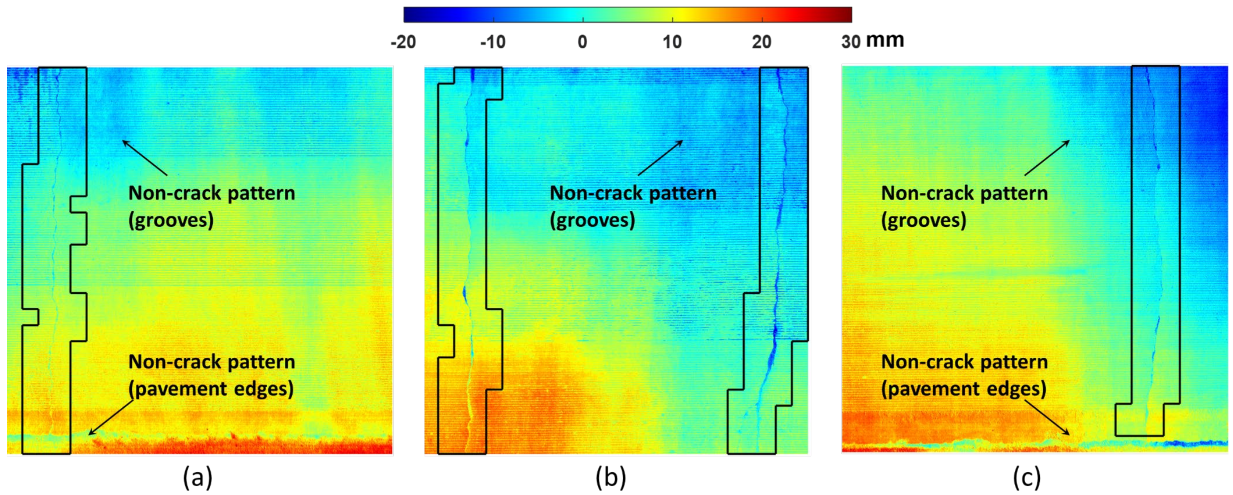


Figure 4-15. Case IV: detected crack maps: (a) Surface 1; (b) Surface 2; and (c) Surface 3.

4.2.4 *Limitations*

Despite the high classification performance as demonstrated in the experimental study, the proposed DCNN methodology has a limitation that it cannot detect shallow cracks, because this range-based methodology relies on the variations in the elevation data to detect the presence of cracks. A few false detection examples on the test datasets 1 and 2, predicted by the No.25 architecture in Case I, are demonstrated in Figure 4-16 (a) and (b), respectively. From Figure 4-16 (a), it is observed that the false negative examples all contain shallow cracks, while the false positive examples are generated due to a lack of non-crack samples with potholes in the training data. Meanwhile, in Figure 4-16 (b), similar issue of false negative detection occurs due to shallow cracks.

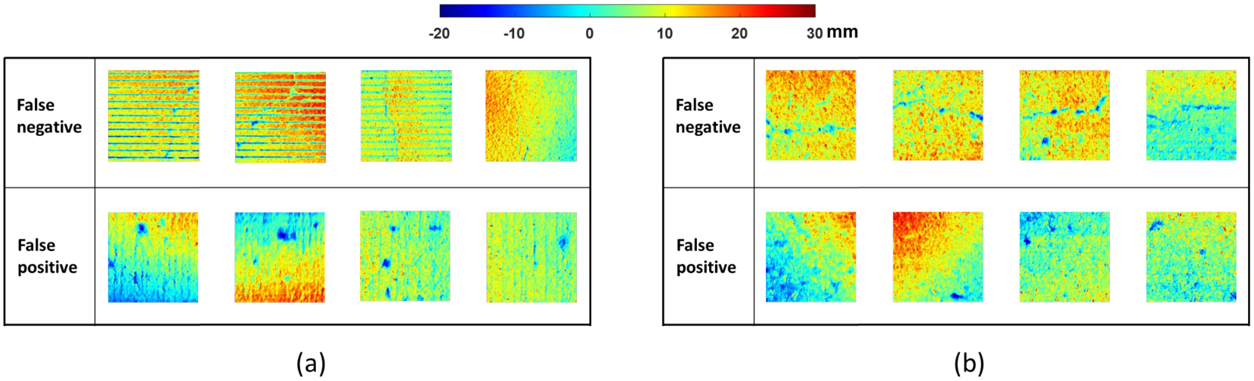


Figure 4-16. False detections on: (a) test dataset 1; and (b) test dataset 2.

4.3 Deep Learning-Based Crack Segmentation

To disambiguate, the terms “Net-1”, “Net-2”, and “Net-3” in this section refer to the DCNN architectures defined in section 3.2.2.

Section 4.3.1 first introduces the data generation process; then, information on the experimental setup is described in section 4.3.2; and, in section 4.3.3, the experimental results and associated discussions are summarized.

4.3.1 *Data Generation*

- Image acquisition and processing

The diversity of the image data used in this study is represented by the following facts: *i*) multiple locations of data acquisition: the image data are acquired through a long-term (over one year) effort from multiple concrete roadways with highly complex surface conditions in the states of Iowa and Alabama; *ii*) high irregularity in crack patterns: the captured cracks include longitudinal and transverse cracks on concrete pavements suffering from the issues including surface variations and grooved patterns; *iii*) various crack orientations through data augmentation: the image data is further processed through augmentation techniques including rotation and mirroring as described in the subsequent section. In total, over 1200 image frames (dimension: 4096×2048×1) are captured. Subsequently, each image frame is cropped into patches (dimension: 256×256×1) through the sliding window technique. As a result, over 4000 image patches containing cracks are selected by trained personnel as the data samples.

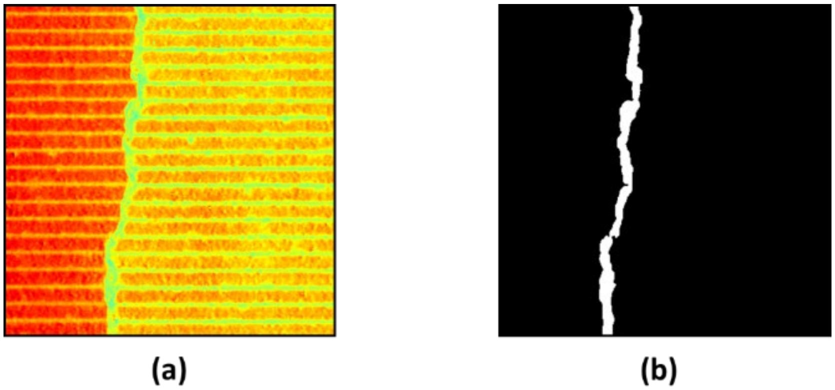


Figure 4-17. An example of the acquired image data: (a) raw range image contaminated with surface variations and grooved patterns; and (b) manually generated ground truth.

- Ground truth generation

The ground truth pixel label map for each image patch is generated through a manual labeling process, and further carefully inspected by trained personnel for quality control. Figure 4-17 illustrates an example of the generated image data. Figure 4-17 (a) shows a raw range image patch contaminated with surface variations and grooved patterns; and, Figure 4-17 (b) illustrates the corresponding ground truth pixel label, where the white pixels are cracks and black ones are non-crack pixels.

- Data augmentation

Data augmentation techniques (section 2.3.3) including rotation (90°, 180°, and 270° counter-clockwise) and mirroring (top-bottom and left-right) are adopted to augment the obtained image patches and the associated ground truth in this study.

- Dataset configuration

Table 4-8 shows the configuration of the acquired datasets, where the image patches are separated into the training, validation, and test datasets following a ratio of 60%:20%:20%. The training dataset is utilized to fit the DCNN model. The validation dataset is evaluated every a few iterations during training to provide a measure on the model fit and indicate whether overfitting occurs. After training, the trained model makes predictions on the test dataset. The metrics on the test dataset provide a cross-comparison between different DCNN models on their segmentation performance.

Table 4-8. Detailed configuration on the image datasets.

Data type	Image dimension	Number of samples		
		Training	Validation	Test
range image	256×256×1	15016	5005	5005

4.3.2 *Experimental Setup*

- Computing hardware and software

The specifications of the data processing computer are as follows: CPU is Intel i7-8750H and GPU is Nvidia GTX 1060 with 6GB RAM. The proposed methodology is implemented in MATLAB R2019a with its deep learning toolbox [143].

- Hyperparameter configuration

In this study, the mini-batch SGD with momentum algorithm (section 2.3.4) is adopted as the optimization technique for training. The associated hyperparameters include the weight decay factor, momentum, learning rate, LReLU factor, mini-batch size, and number of epochs. It is noted that, upon training different architectures, the same hyperparameter values are adopted in this study to provide a basis for comparison. The hyperparameter values are as follows: weight decay factor = 0.0003; momentum = 0.9; initial learning rate = 0.01; learning rate drop factor = 0.8; LReLU factor = 0.01; mini-batch size = 10; number of epochs = 10.

- Parameter initialization

The learnable parameters are initialized based on the settings in section 2.3.4.

4.3.3 *Results and Discussions*

Two experimental cases are performed in this study. Case I is designed to investigate the influences from different network depths and residual connections on crack segmentation performance. From Case I, an optimal architecture among the twelve proposed DCNNs that yields the highest performance measures is selected. Then, in Case II, the performance of the optimal architecture is further demonstrated on three roadway images.

- Case I: comparison on the segmentation performance

As introduced in section 3.3, this study proposes six encoder-decoder networks denoted as Net-1 through 6 with gradually increased network depths; from Net-1 through 6, the number of residual connections is also increased from 1 to 6, as listed in Table 3-1 and illustrated in Figure 3-12 and Figure 3-13. Besides, to isolate the impacts from increasing the network depth and from adding residual connections, another six architectures denoted as Net-1A through 6A are also designed as "plain" counterparts; these variants have the same layer configurations as their originals except they do not employ any residual connection.

Besides, CrackNet II [30], a DCNN developed for roadway crack segmentation with range images, is adopted in Case I for comparison. Thus, in total, thirteen architectures are trained and tested on the same datasets under the same hyperparameter configurations. Five metrics including Precision, Recall, F1, IOU, and BF score are evaluated on the validation dataset and test dataset, respectively, with their mean values tabulated in Table 4-9. Meanwhile, the highest value in each type of metrics is highlighted in bold font. From the "validation metrics" section in Table 4-9, it can be seen Net-4 has the highest values in mean F1, IOU, and BF score among all cases, indicating the best model fit. Net-4 also yields the highest values in the mean F1, IOU, and BF score calculated on the test dataset. The metrics are further illustrated in Figure 4-18 to compare the performance among different architectures. Figure 4-18 (a-e) shows the mean values of the Precision, Recall, F1, IOU, and BF score evaluated on the test dataset, respectively. The horizontal axis refers to different architectures, and the vertical axis represents percentage values. The following observations can be made from Figure 4-18:

i) In all five metrics, the networks with residual connections (blue color) yield higher values than their "plain" counterparts without residual connections (red color), especially in deeper architectures (e.g., Net-6 vs. Net-6A);

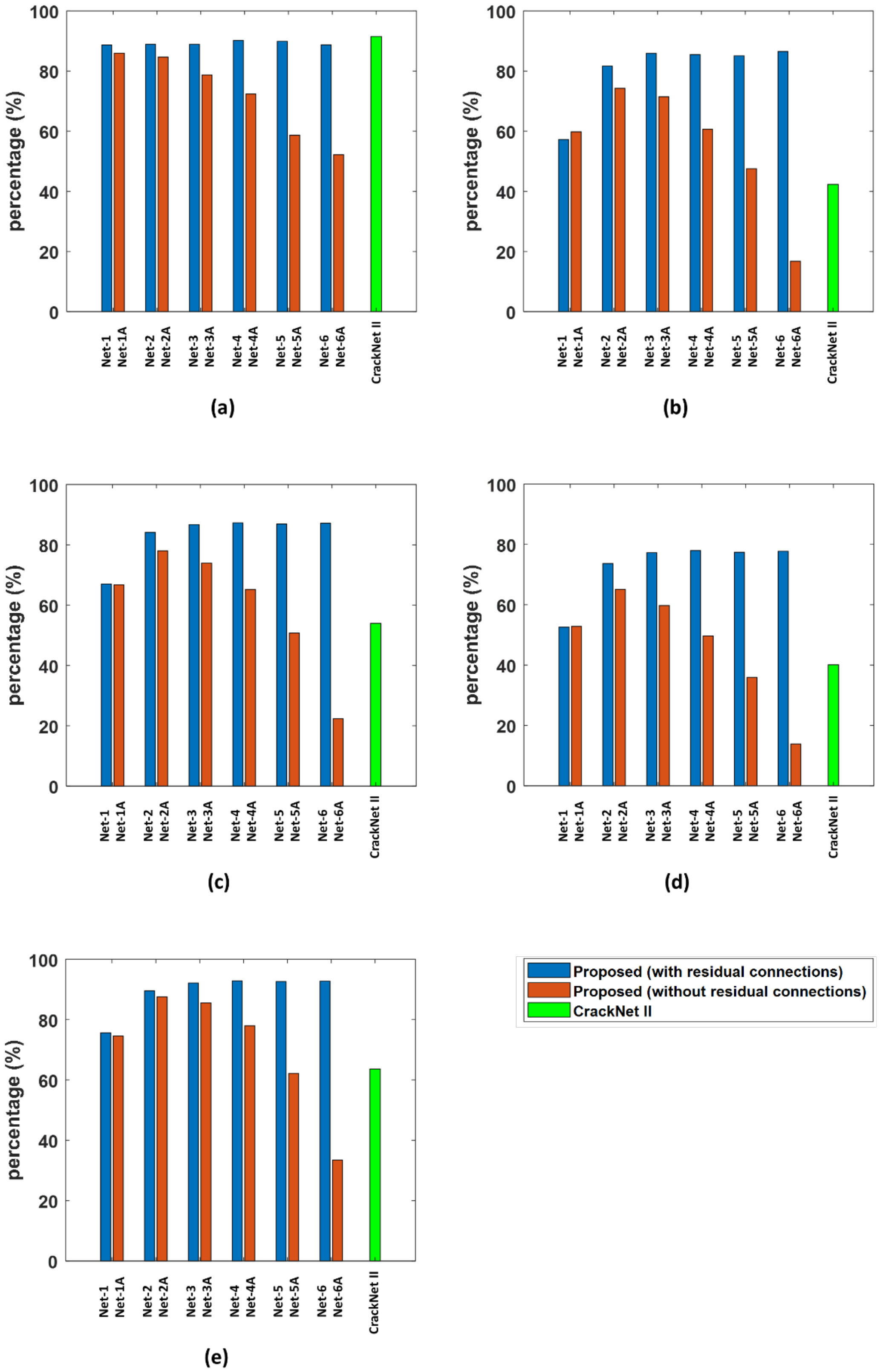
ii) From Net-1A through Net-6A (without residual connections), the network depth is increased from 6 to 16 layers (see Table 3-1). Meanwhile, their metrics values first reach the peak at Net-2A then deteriorate. The mean F1, IOU, and BF score drop by over 50% from Net-2A to Net-6A. It is observed that deeper architectures (Net-2A through 6A) without residual connections lead to worse performance on the test dataset;

iii) From Net-1 through Net-6 (with residual connections), while architecture becomes deeper, the corresponding metrics first increase then plateau at the highest values. It is thus demonstrated that using residual connections is beneficial in that it avoids performance degradation in deeper architectures by providing low-level information to the decoder;

iv) Judging by the mean F1, IOU, and BF score values, Net-4 results in the highest performance on the test dataset, thus it is selected as the optimal architecture; by comparing it with CrackNet II (green color), it can be seen that CrackNet II only exceeds marginally in the mean Precision value by 1.4%; however, Net-4 outperforms CrackNet II on the mean Recall, F1, IOU, and BF score by 43.2%, 33.4%, 37.9%, and 29.3%, respectively. Thus, the segmentation performance of the proposed network (i.e., Net-4) on range image data is validated through a comparison with other semantic DCNNs.

Table 4-9. Case I: performance metrics.

Index	Net name	Validation metrics (mean percentage values)					Testing metrics (mean percentage values)					Time	
		Precision	Recall	F1	IOU	BF score	Precision	Recall	F1	IOU	BF score	Training (min)	Testing (sec)
1	Net-1	88.9	57.4	67.2	52.8	75.8	88.7	57.2	67.0	52.6	75.6	817.5	105.9
2	Net-2	88.9	81.8	84.2	73.8	89.7	88.9	81.7	84.1	73.7	89.6	872.0	111.5
3	Net-3	88.8	85.9	86.7	77.2	92.1	88.9	85.9	86.7	77.2	92.2	818.0	122.3
4	Net-4	90.1	85.4	**87.2**	**77.9**	**92.8**	90.1	85.5	**87.3**	**78.0**	**92.9**	934.9	134.5
5	Net-5	89.9	85.1	87.0	77.5	92.6	89.8	85.1	86.9	77.4	92.7	947.6	162.3
6	Net-6	88.7	**86.5**	87.1	77.7	92.7	88.7	**86.5**	87.1	77.7	92.8	1473.7	162.1
7	Net-1A	86.1	59.8	66.9	53.0	74.9	85.9	59.8	66.8	52.8	74.6	833.2	105.5
8	Net-2A	84.5	74.5	78.1	65.2	87.7	84.6	74.3	78.0	65.1	87.6	850.2	108.1
9	Net-3A	78.5	71.6	74.0	59.7	85.6	78.7	71.5	73.9	59.7	85.5	858.4	136.2
10	Net-4A	72.2	60.7	65.1	49.6	78.1	72.4	60.7	65.2	49.7	78.0	908.2	149.6
11	Net-5A	58.3	47.3	50.5	35.6	62.1	58.6	47.5	50.8	35.9	62.2	940.5	155.0
12	Net-6A	52.1	16.3	21.9	13.5	33.0	52.2	16.7	22.4	13.8	33.5	1521.6	180.1
13	CrackNet II	**91.5**	42.8	54.5	40.6	64.1	**91.5**	42.3	53.9	40.1	63.6	2127.2	244.8



(a)

(b)

(c)

(d)

(e)

Figure 4-18. Case I: performance metrics on the test dataset: (a) Precision; (b) Recall; (c) F1; (d) IOU; and (e) BF score.

In addition to showing the mean values of each type of metrics, the histograms of the metrics evaluated on the test dataset (which contains more than 5000 range images) are also displayed in Figure 4-19. In this figure, each plot is arranged such that the five metrics including Precision, Recall, F1, IOU, and BF score are illustrated from left to right; in addition, the histogram generated by CrackNet II is also illustrated in this figure; the horizontal axis of each plot refers to the percentage value of each type of metrics, and the vertical axis represents the frequency counts. Both axes are plotted in the same range and scale. From Figure 4-19, it is clear that from Net-2A through Net-6A, the histogram of each type of metrics gradually shifts, resulting in a larger dispersion and smaller mean value; such a phenomenon indicates performance deterioration caused by increasing the network depth. Regarding the effect of residual connections, adding residual connections results in a marginal difference in shallow architectures (i.e., Net-1 vs. Net-1A). However, as the architecture evolves into a deeper layout from Net-1 to Net-6, adding residual connections prevents the performance degradation issue which occurs among the architectures without residual connections. Again, it can be observed the optimal architecture Net-4 outperforms CrackNet II, in that the histograms of Net-4 for the Recall, F1, IOU, and BF score have smaller spreads and higher mean values.

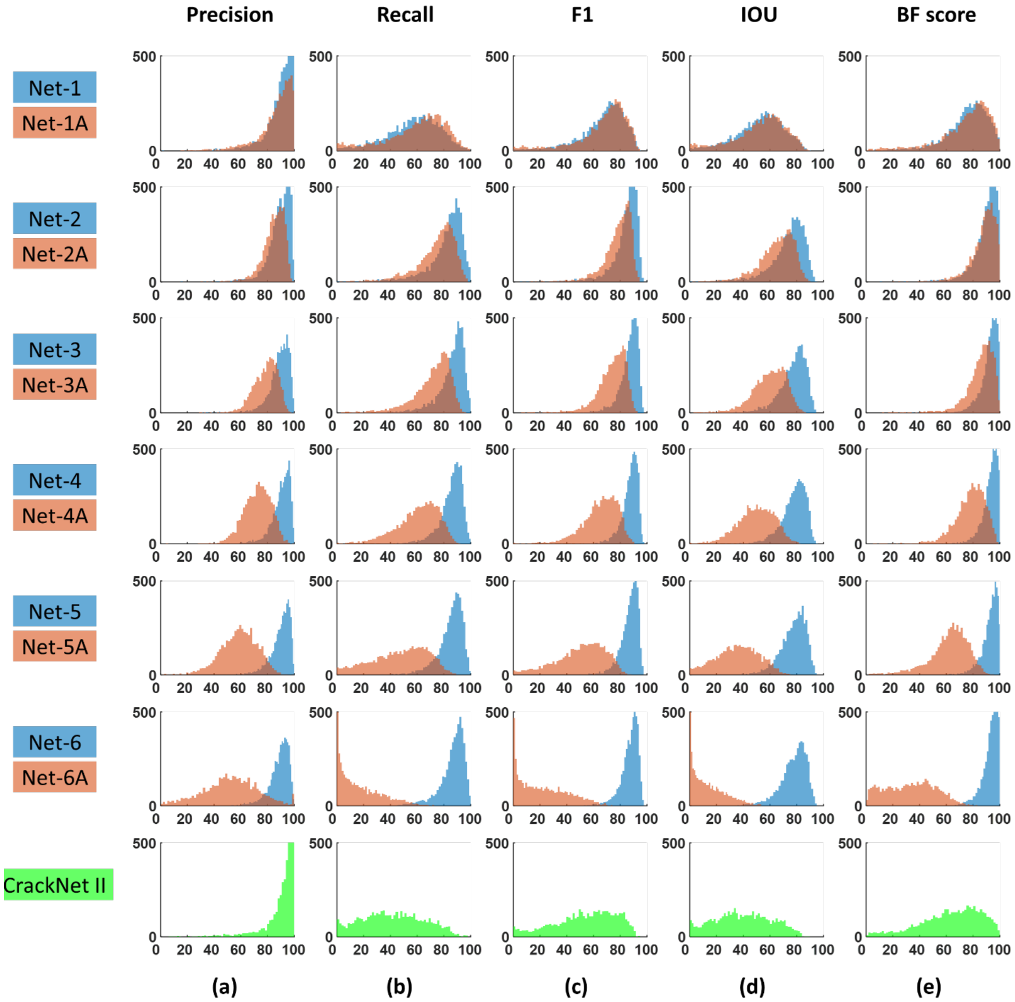


Figure 4-19. Case I: histograms of the performance metrics on the test dataset: (a) Precision; (b) Recall; (c) F1; (d) IOU; and (e) BF score.

Testing results on twelve crack images which are contaminated with surface variations and grooves are illustrated in Figure 4-20 to demonstrate the segmentation performance by different networks. In Figure 4-20, the raw range image input, pixel label ground truth, and segmentation results are displayed from top to bottom. Meanwhile, the mean percentage values of F1, IOU, and BF score are labelled on top of each corresponding result. It is observed that Net-1, Net-1A, and CrackNet II yield some false positive detections by misidentifying the grooves as cracks. By increasing the network depth, such a misidentification issue vanishes for the proposed DCNNs. Again, it is shown that Net-4 yields the highest segmentation performance on these image samples. Furthermore, the influence of residual connection is also revealed in this figure by comparing each network pair (e.g., Net-4 vs. Net-4A). It can be seen that the networks with residual connections (Net-2 through 6) can yield higher metrics values on almost all twelve samples than their "plain" counterparts. Similar as the general trends observed in Figure 4-19, from Net-2A through Net-6A, the segmentation performance keeps deteriorating; on the contrary, due to the existence of residual connections, Net-2 through Net-6 yield very consistent crack segmentation performance under the disturbance of grooves.

Regarding the efficiency of the proposed architectures, the training and testing time are listed in Table 4-9 and illustrated in Figure 4-21. Generally, from Net-1 through Net-5, as the network becomes deeper, the training time has little variation; however, in Net-6, the time cost for training is drastically increased by over 70%, indicating a deteriorated training efficiency. The same trend can be observed among Net-1A through 6A as well. Also, using residual connections yields very marginal influence on the training time. Regarding the testing efficiency, the deeper architectures require a longer time for testing; for example, the testing time is increased by 70% from Net-1A to 6A. Meanwhile, as can be observed, using residual connections leads to up to 12% variations in the testing time, and its impact on the testing efficiency is not consistent. Additionally, from Figure 4-21, it is clear that the optimal architecture Net-4 is more efficient than CrackNet II in that Net-4 only consumes 44% and 55% of the time cost for training and testing, respectively, by CrackNet II.

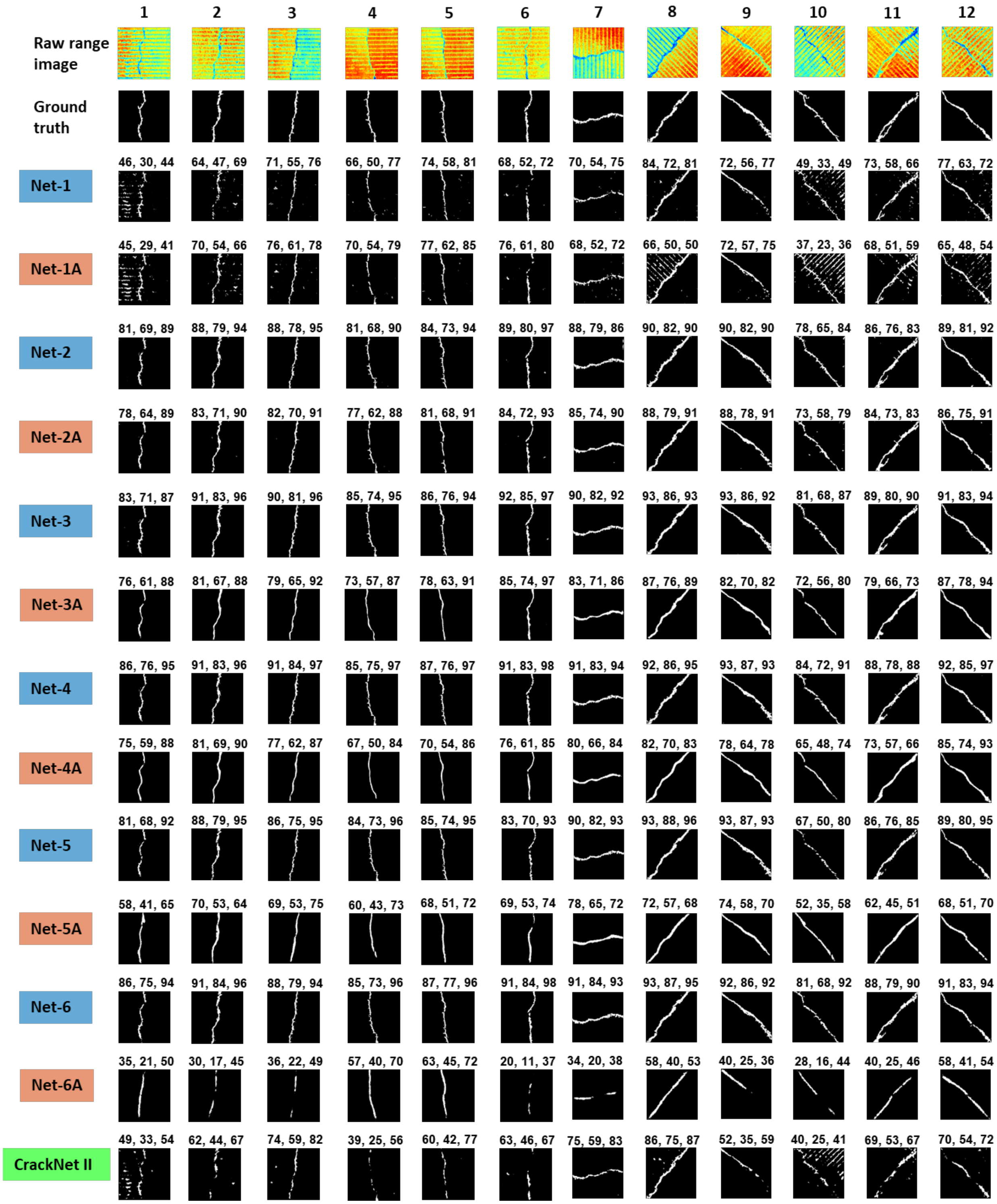


Figure 4-20. Case I: Illustrative examples of the crack segmentation performance (the F1, IOU, and BF score values are displayed at the title of each prediction).

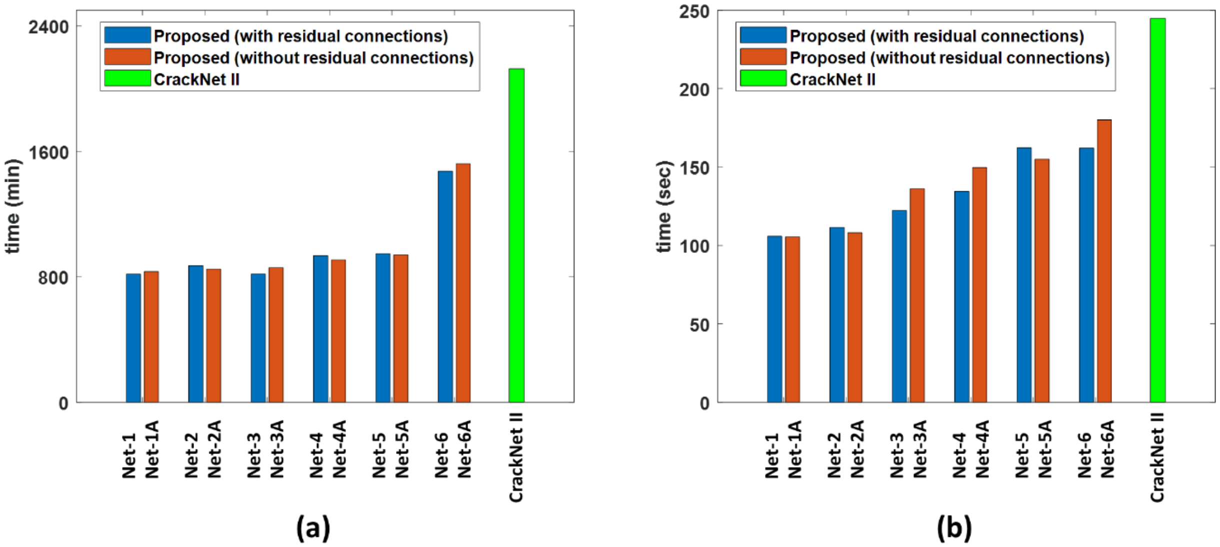


Figure 4-21. Case I: network efficiency: (a) training time; and (b) testing time.

- Case II: performance demonstration on concrete roadway images

This section further demonstrates the segmentation performance of the optimal architecture, Net-4, on three large roadway surfaces (dimension: 2048×2048×1). All three surfaces contain different levels of surface variations and grooved patterns, reflecting the real-world complexities. A pixel-level crack map is generated for each roadway surface, following the framework described in section 3.3.2. It is noted the range image data in Case II did not participate in the training in Case I. Figure 4-22 (a-c) illustrates the segmentation results on Surfaces 1, 2, and 3, respectively. In Figure 4-22, the raw range image, ground truth, and predicted crack map of each surface are illustrated from top to bottom, respectively. Three metrics including F1, IOU, and BF score are employed to provide a quantitative measure on the segmentation performance on each image surface, as listed in Table 4-10. The F1, IOU, and BF score values on all three images are above 80%, 70%, and 90%, respectively, indicating very high segmentation performance by Net-4. Thus, it is demonstrated the proposed methodology can achieve accurate and robust crack segmentation performance on laser-scanned roadway range images, in which the cracks are contaminated by surface variations and grooved patterns.

Table 4-10. Case II: performance metrics.

Image name	Data type	Performance metrics (%)		
		F1	IOU	BF score
Surface 1	range image	87.1	77.1	98.5
Surface 2	range image	84.8	73.6	96.9
Surface 3	range image	83.6	71.9	92.4

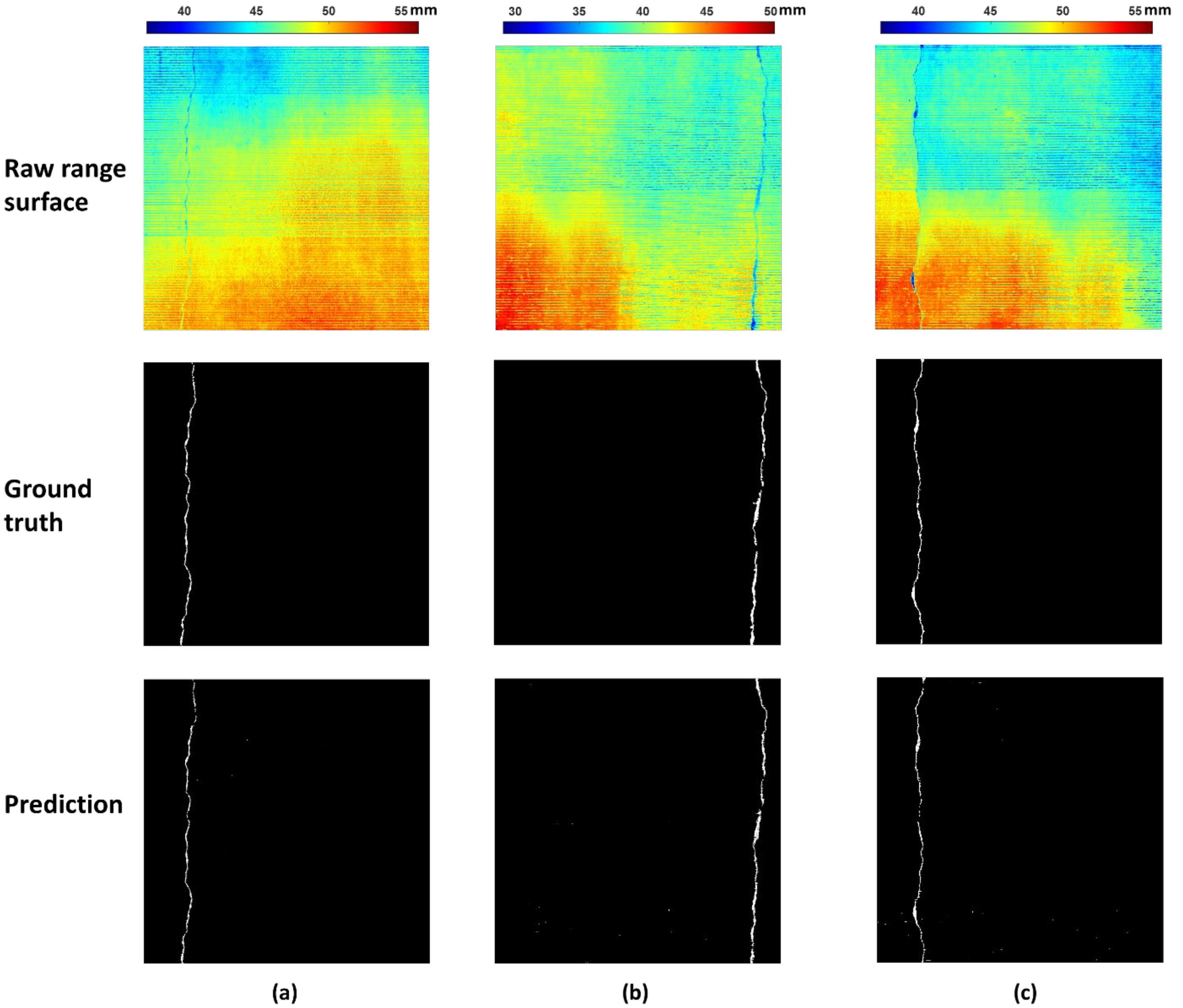


Figure 4-22. Case II: Predicted crack maps on roadway images: (a) Surface 1; (b) Surface 2; and (c) Surface 3.

4.3.4 *Limitations*

The proposed DCNN-based methodology may suffer from the issue of shallow cracks in the range image data, as shown in Figure 4-23. This figure illustrates ten samples containing shallow cracks, where the raw range image, ground truth, and predicted crack map are displayed from top to bottom, respectively. Net-4 trained in Case I is utilized for testing the raw range images. The percentage values of three metrics including F1, IOU, and BF score are calculated and labelled on top of each predicted crack map. By both judging from the low metrics values and from graphically comparing the predicted crack map vs. ground truth, it can be observed that the performance of the proposed DCNN deteriorates on images containing shallow cracks, represented by false negative detections.

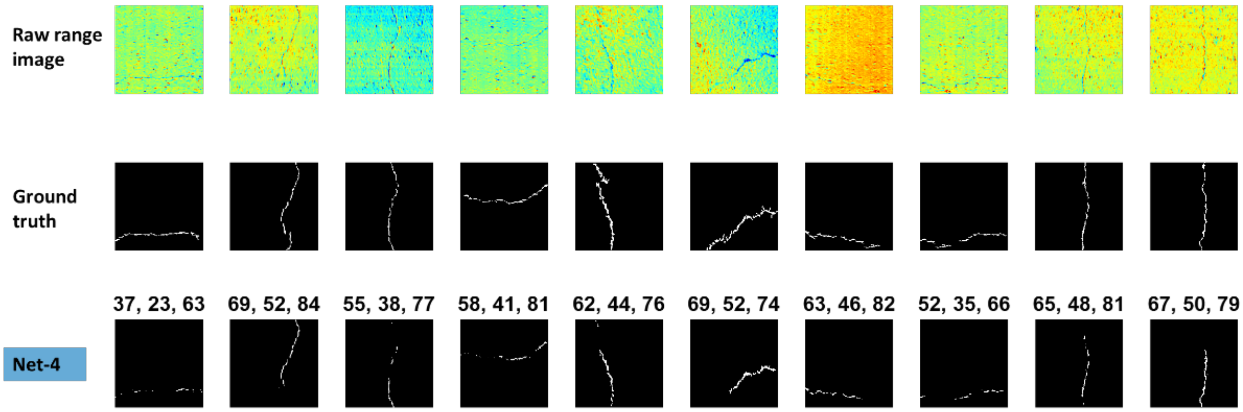


Figure 4-23. Performance deterioration due to shallow cracks.

4.4 Deep Learning-Based Data Fusion for Crack Classification

In this experimental study section, first, the data acquisition and data generation process are introduced in section 4.4.1; then, information on the experimental setup is provided in section 4.4.2; in section 4.4.3, two experimental cases are performed to evaluate the network performance with different image types and architectures.

4.4.1 *Data Generation*

- Image acquisition and processing

In this study, the raw image range and intensity data are collected by the laser imaging system over a one-year period on multiple concrete roadways. The acquired raw range images are further processed by trained personnel using the proposed image pre-processing technique introduced in section 3.1.4, to generate filtered range images which are free from surface variations, scanning noises, and non-crack patterns. A sliding window technique as used in [39,93] is applied to crop all the image data into many patches which have smaller sizes to reduce computational cost. Also, the fused raw image patches are created using the raw range and intensity image pairs by the procedure described in section 3.4.2. As a result, four types of image patches including raw range, raw intensity, filtered range, and fused raw images are prepared for analysis. The dimension of each image patch is 256×256×1 for the raw range, raw intensity, and filtered range image, but 256×256×2 for the fused raw image.

- Ground truth generation

Each image patch is categorized with a "crack" or "non-crack" label by trained personnel.

- Data augmentation

Meanwhile, to effectively increase the number of samples (i.e., image patches) in each type of image, data augmentation techniques including rotation (counterclockwise 90°, 180°, and 270°) and mirroring (left-right and up-down) are performed. Data augmentation is an effective approach to reduce overfitting and improve generalization by increasing the number of image data through label-preserving transformations [92,96]. In total, 30000 crack samples (including augmented samples) and 30000 non-crack samples are generated for each type of image data; the amount of crack and non-crack samples is kept the same to avoid the issue of class imbalance [103].

- Dataset configuration

The image samples are further separated into three datasets including the training dataset, validation dataset, and test dataset following a ratio of 60%:20%:20%. The training dataset is used to fit the DCNN model. The validation dataset is evaluated during training to provide a measure on the goodness of model fit and indicate if overfitting occurs. After training is completed, the network is tested on the test dataset to evaluate the model ability to generalize and adapt to a new dataset. Detailed configuration on the image dataset for each type of image data is tabulated in Table 4-11.

The above data augmentation and dataset generation process is repeated for each type of image data. Thus, each type of image forms a set of training, validation, and test datasets.

Table 4-11. Detailed configuration on the image dataset.

	Training	Validation	Test	Total
Crack	18000	6000	6000	30000
Non-crack	18000	6000	6000	30000
Total	36000	12000	12000	60000

4.4.2 *Experimental Setup*

Two cases are performed in this section. Case I investigates the impacts on DCNN-based roadway crack classification by heterogeneous image data including raw range, raw intensity, filtered range, and fused raw image; Case II further compares the performance of two architectures using the fused raw image data, upon which the one leading to better classification performance is determined.

- Computing hardware and software

On performing all the experiments, the same computing device is used to isolate the influence from different devices. The specifications of the computer are as follows: CPU: Intel Xeon E5-2630 @2.20 GHz; GPU: Nvidia Quadro M4000 with 8GB memory. MATLAB [143] and its deep learning tool box are utilized to implement the proposed DCNNs.

- Hyperparameter configuration

The mini-batch stochastic gradient-descent (SGD) with momentum [99] is adopted in this study as the optimization technique for training. The hyperparameters involved in this algorithm are specified as follows. The momentum is set as 0.9. The weight decay is set as 0.0003. The initial learning rate and the corresponding learning rate drop factor are selected as 0.01 and 0.8, respectively; the decay period of the learning rate is set as 2 epochs; that is, the learning rate decays by multiplying 0.8 in every 2 epochs. One epoch is a full pass of the entire image dataset through mini batches. The mini-batch size is 60; that is, in each iteration, 60 images which have not yet participated in the training in the current epoch are randomly chosen for updating. According to the data generation process (see section 4.4.1), a total number of 36000 image samples are prepared for training (see Table 4-11), resulting in 600 iterations in an epoch with each iteration containing 60 samples. Meanwhile, the validation dataset is evaluated every half an epoch. The statistics evaluated on the validation dataset at the end of training are used to interpret the goodness of model fit.

- Parameter initialization

The weights in the convolutional layers and fully connected layers are initialized by the Glorot initializer [100], which independently samples the weights from a Gaussian distribution with zero mean and a variance based on the dimension of the weights. The biases are initialized as 0. The scale and shift factor in each batch normalization layer are initialized as 1 and 0, respectively.

4.4.3 *Results and Discussions*

In this section, two experimental cases are performed. Case I compares the performance of a series of DCNN architectures with heterogeneous image data, determining the type of image data and the associated architecture that yield the highest classification performance. Case II further determines the better heterogeneous image fusion strategy with DCNN through a comparison.

- Case I: comparison on network performance by heterogeneous image data

Net-A (see section 3.4.3), which is designed for single-channel image input, is trained and tested on three types of image data including the raw range, raw intensity, and filtered range image datasets. Furthermore, the fused raw image data is utilized by Net-B (see section 3.4.3) which has the similar layout as Net-A except it takes dual-channel image input. Comparison on the performance metrics demonstrates the optimal form of image data to achieve the best classification performance.

A DCNN architecture [39] is employed as a benchmark architecture to further evaluate the effects of heterogeneous image data. This benchmark architecture is also utilized in some other applications such as [74,152] for comparison. It is noteworthy that the benchmark architecture originally utilizes raw intensity images for classification; in this study, it is trained and tested not only on raw intensity images, but also on raw range and filtered range images for analysis.

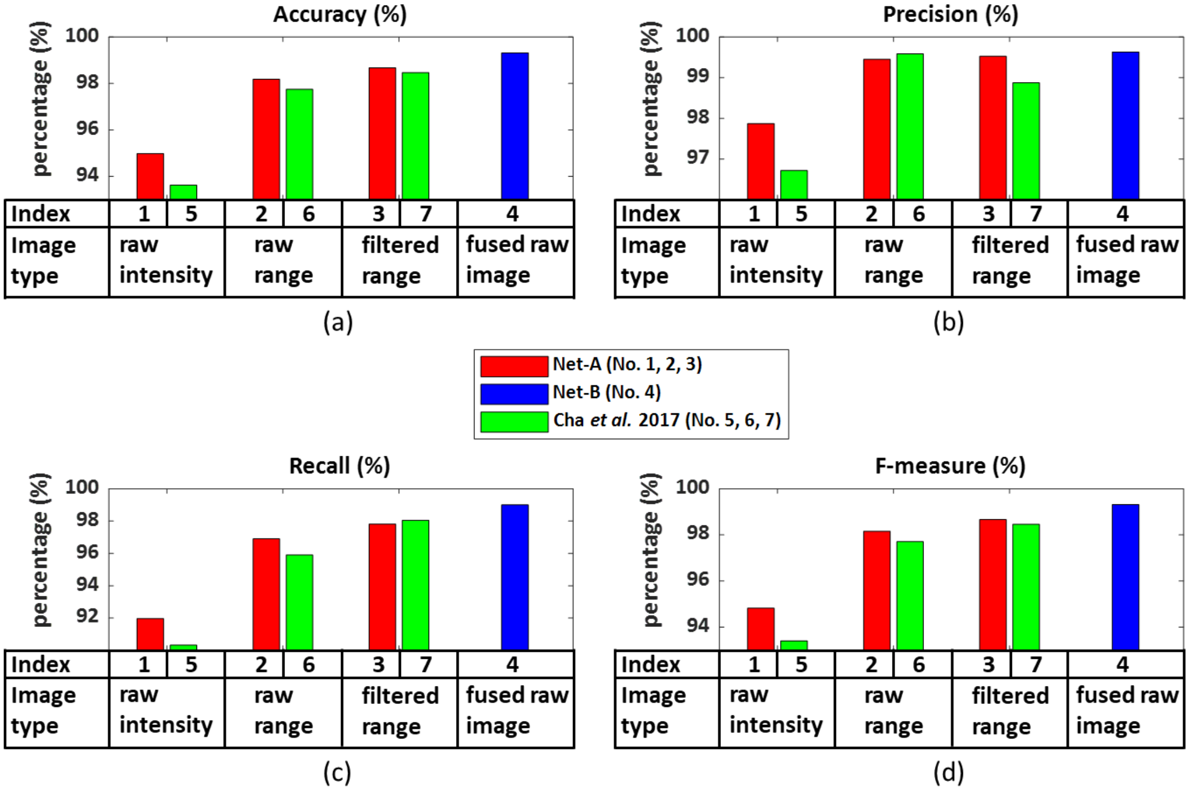


Figure 4-24. **Case I: Performance metrics on the test dataset: (a) Accuracy; (b) Precision; (c) Recall; and (d) F1.**

The performance metrics including Accuracy, Precision, Recall, and F1 on the test dataset are illustrated in Figure 4-24 (a), (b), (c), and (d), respectively. Meanwhile, the detailed statistics on the performance metrics are tabulated in Table 4-12. Case I consists of seven subcases (No. 1-7 in Table 4-12). In the subcases No. 1, 2, 3 in Table 4-12, Net-A is trained and tested on the raw intensity, raw range, and filtered range image datasets, respectively; No. 4 corresponds to Net-B trained and tested on the fused raw image dataset; the subcases No. 5, 6, and 7 utilize the benchmark architecture for training and testing on the raw intensity, raw range, and filtered range image datasets, respectively. In all four plots of Figure 4-24, the horizontal axis refers to the subcase indexes as defined in Table 4-12, which are further grouped by the image types, including raw intensity, raw range, filtered range, and fused raw image . Several observations on this figure are summarized below:

i) It is evident that using intensity image data for training and testing results in the lowest values on all four metrics shown in Figure 4-24. The Recall values for the subcases using intensity images [No. 1 and 5 in Figure 4-24 (c)] are especially lower than others, indicating a much higher false-negative rate. As will be demonstrated later in Figure 4-26, these false-negatives are induced by the issue from low contrasts between cracks and non-crack regions in the intensity images, which is the main reason leading to the worst performance;

ii) Based on the comparisons on the following subcases "No. 1 vs. No. 2" and "No. 5 vs. No. 6", it is demonstrated that using raw range image instead of intensity image can effectively improve the classification performance. Given the same architecture used, the F1 values on the test dataset are improved by 3% (Net-A in No. 1 and 2) and 4% (the benchmark in No. 5 and 6), respectively, due to the use of raw range images instead of raw intensity images. Similar level of improvement on the other metrics is also observed. Therefore, based on the data acquired in this study, it can be concluded that using raw range images is much better than using raw intensity images on DCNN-based roadway crack classification;

iii) To evaluate the effect of image pre-processing, comparisons between the subcases "No. 2 vs. No. 3" and "No. 6 vs. No. 7" are performed. It can be observed that in these two comparisons, utilizing filtered range images instead of raw images results in 0.5% and 0.7% improvements on the F1 values on the test dataset, respectively. It is demonstrated that image pre-processing can improve the network performance on DCNN-based roadway crack classification by a noticeable margin. Nevertheless, considering such an improvement is achieved through an additional image pre-processing procedure which may require a certain level of expertise, it is still preferable to directly use the raw range image for real-world applications;

iv) It is clearly demonstrated in Figure 4-24 that using fused raw image data yields the highest values in all four metrics. For example, by comparing the subcases between No. 2 (using raw range image) and No. 4 (using fused raw image), around 1.2% improvement on the F1 value on the test dataset are observed. Meanwhile, it can be concluded that using fused raw image leads to a more significant improvement on the classification performance than using filtered range images, where the F1 value on the test dataset in No. 4 (using fused raw image) is 0.7% higher than that in No. 3 (using filtered range images). Unlike the image pre-processing procedure, the heterogeneous image fusion process is a more intuitive and straightforward approach to process the original raw image data, which only requires acquisition of image range and intensity data. Therefore, heterogeneous image fusion can be a more practical approach to improve the performance of a DCNN model under real-world scenarios;

v) The proposed DCNN architecture yields similar or better performance in all the metrics comparing to the benchmark. Furthermore, the above observations on the effects of utilizing the raw intensity, raw range, and filtered range image data are mostly consistent between the proposed and benchmark DCNNs. It is noted that the benchmark DCNN shows a lower Precision value when using the filtered range image than the raw range image [see Figure 4-24 (b)], which is different than the proposed DCNN.

Table 4-12. Detailed statistics on the performance metrics.

Index	Architecture	Image type	Validation				Testing				Time	
			Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	Training (min)	Testing (sec)
1	Net-A	raw intensity	95.5	98.2	92.6	95.3	95.0	97.9	92.0	94.8	121.9	21.4
2	Net-A	raw range	98.2	99.3	97.2	98.2	98.2	99.5	96.9	98.2	102.6	25.4
3	Net-A	filtered range	98.6	99.4	97.7	98.6	98.7	99.5	97.8	98.7	129.1	25.3
4	Net-B	fused raw image	99.2	99.4	99.1	99.2	99.3	99.6	99.0	99.3	173.2	38.2
5	Cha *et al*. 2017	raw intensity	93.9	96.8	90.8	93.7	93.6	96.7	90.3	93.4	96.5	21.3
6	Cha *et al*. 2017	raw range	97.8	99.5	96.1	97.7	97.8	99.6	95.9	97.7	96.0	21.4
7	Cha *et al*. 2017	filtered range	98.3	98.4	98.1	98.3	98.5	98.9	98.1	98.5	125.9	21.2
8	Net-C	fused raw image	99.0	99.5	98.4	99.0	99.0	99.6	98.3	99.0	185.5	44.0

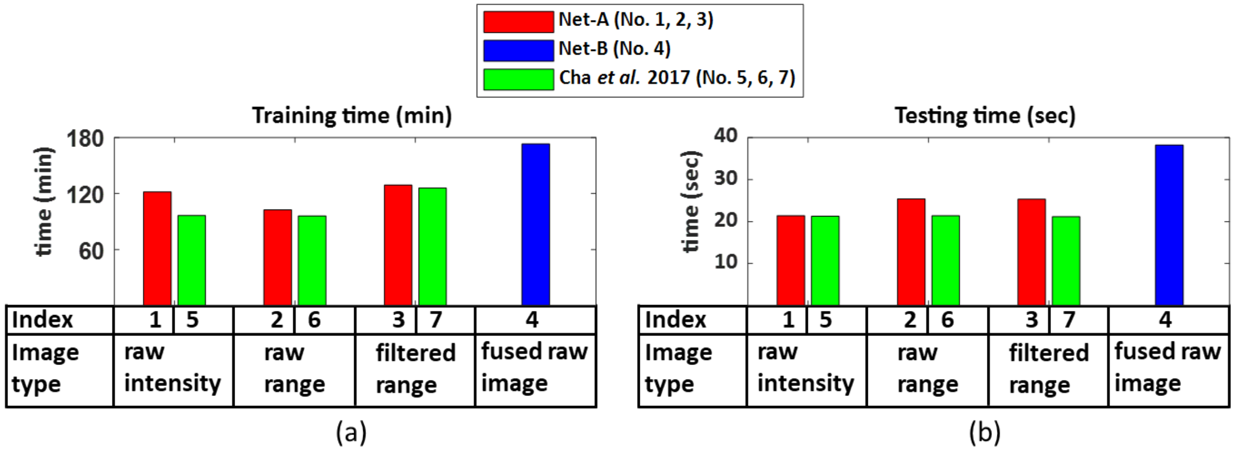


Figure 4-25. Case I: Performance metrics: (a) training time; and (b) testing time.

Regarding the efficiency, Figure 4-25 illustrates the training time and testing time for the subcases in Case I. The horizontal axis in this figure is the same as specified in Figure 4-24; the vertical axis refers to the training and testing time for Figure 4-25 (a) and (b), respectively. It can be observed that, for the proposed DCNNs, using raw intensity, raw range, and filtered range images result in a very similar time consumption, but they are all more efficient than the subcases with fused raw images. The benchmark DCNN requires similar amount of time as the proposed. In comparison, the training and testing time are increased by around 30% and 50% from Net-A (No. 1-3) to Net-B (No. 4). This phenomenon may be caused by the increase in the volume of the input image data due to heterogeneous image fusion.

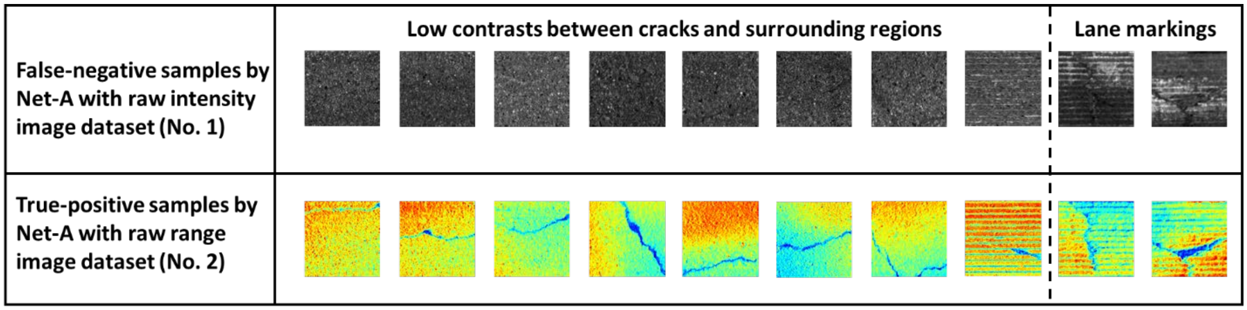

Low contrasts between cracks and surrounding regions
Lane markings
False-negative samples by Net-A with raw intensity image dataset (No. 1)
True-positive samples by Net-A with raw range image dataset (No. 2)

Figure 4-26. Case I: Crack samples misidentified by Net-A with raw intensity image dataset (No. 1) but correctly identified by Net-A with raw range image dataset (No. 2).

To further demonstrate the impact from different image datasets on the network performance, some prediction results are illustrated in Figure 4-26 and Figure 4-27, respectively. Figure 4-26 demonstrates some false-negative samples predicted by Net-A which is trained and tested on the raw intensity image dataset (No. 1 in Table 4-12). The false negatives shown in Figure 4-26 suffer from low contrasts between cracks and non-crack regions and contaminations such as lane markings. On the contrary, the cracks are very distinguishable in the corresponding range images because of the elevation change. By utilizing the range image dataset, the same DCNN architecture (No. 2) can correctly detect and classify these image samples previously misidentified when using intensity images (No. 1).

Meanwhile, some false-negative samples predicted by Net-A with the raw range image dataset (No. 2) are illustrated in Figure 4-27. As can be observed from this figure, most false-negative samples contain shallow cracks which are further contaminated with grooved patterns. Such shallow cracks, however, are correctly identified by Net-B with the fused raw image dataset (No. 4) by integrating the information from the corresponding intensity images. It is thus demonstrated that by exploiting the spatial co-registration feature of cracks offered by heterogeneous data fusion of range and intensity images, the classification accuracy can be effectively improved.

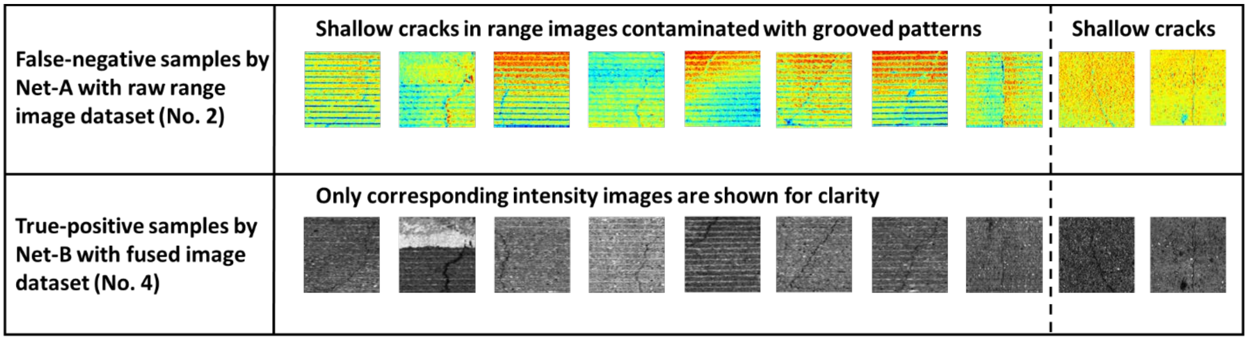

False-negative samples by Net-A with raw range image dataset (No. 2)
Shallow cracks in range images contaminated with grooved patterns
Shallow cracks
True-positive samples by Net-B with fused image dataset (No. 4)
Only corresponding intensity images are shown for clarity

Figure 4-27. **Case I: Crack samples misidentified by Net-A with raw range image dataset (No. 2) but correctly identified by Net-B with fused raw image dataset (No. 4).**

Furthermore, in Case I, an example of crack detection on a concrete roadway image (denoted as "Surface 1") is illustrated in Figure 4-28 to demonstrate the effects of different types of image data on DCNN performance. In Figure 4-28 (a), (b), and (c), the raw intensity, raw range, and filtered range images of Surface 1 are displayed; the fused raw image data is generated by combining Figure 4-28 (a) and (b), and is not shown here. Surface 1 contains two major cracks propagating along the transverse direction of the roadway, and many non-crack patterns (surface variations and grooves). As can be observed in Figure 4-28 (a), some cracked regions in the raw intensity image have the issue of low intensity contrast between the cracks and surroundings; similarly, in Figure 4-28 (b), shallow cracks exist in a few regions in the raw range image.

The crack maps based on the raw intensity, raw range, and filtered range images are generated by the trained Net-A (No. 1, 2, and 3 in Table 4-12, respectively); and, the crack map based on the fused raw image is generated by the trained Net-B (No. 4 in Table 4-12). Net-A and Net-B are quite similar in terms of their layouts, only except that their input layer and the first convolutional layer are different. Thus, the major impact factor on the detection performance is the different types of image data used for DCNN analysis. Figure 4-28 (d-g) illustrate the crack maps detected from the raw intensity, raw range, filtered range, and fused raw image data, respectively, where the blue, red, and black patches indicate true-positive, false-negative, and false-positive detections. It is noted that the crack maps are all overlaid with the same raw range image for cross comparison. The detailed precision-recall statistics are tabulated in Table 4-13.

It can be seen from Figure 4-28 (d-g) and Table 4-13 that, the crack detection result based on the raw intensity image yields the worst performance (e.g., F1 = 88.5%), including many false-negative detections due to the low contrast issue. By using raw range or filtered range image, the corresponding F1 values are increased to 95.2% and 94.0%, respectively. Meanwhile, comparison between the detection results using raw range image vs. filtered range image suggests that applying image pre-processing on the raw range image of Surface 1 does not yield higher classification performance in this example. Instead, by using the strategy proposed in this study to fuse the raw intensity and raw range image, the F1 values on Surface 1 are effectively increased by 10.4% (using raw intensity vs. fused raw image) and 3.7% (using raw range vs. fused raw image). As can be observed by comparing the differences between Figure 4-28 (g) and Figure 4-28 (d, e, f), the use of fused raw image data for analysis helps eliminate many false-negative and false-positive detections by cross-domain feature correlation and thus leads to the highest classification performance. Through this example, it is again demonstrated that the proposed data fusion strategy can effectively improve the DCNN performance on roadway crack detection.

Table 4-13. Precision-recall statistics of crack detection on Surface 1.

Index	Architecture	Image type	Testing metrics on Surface 1			
			Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
1	Net-A	raw intensity	96.2	98.7	80.2	88.5
2	Net-A	raw range	98.3	97.8	92.7	95.2
3	Net-A	filtered range	97.9	98.9	89.6	94.0
4	Net-B	fused raw image	**99.6**	**100.0**	**97.9**	**98.9**

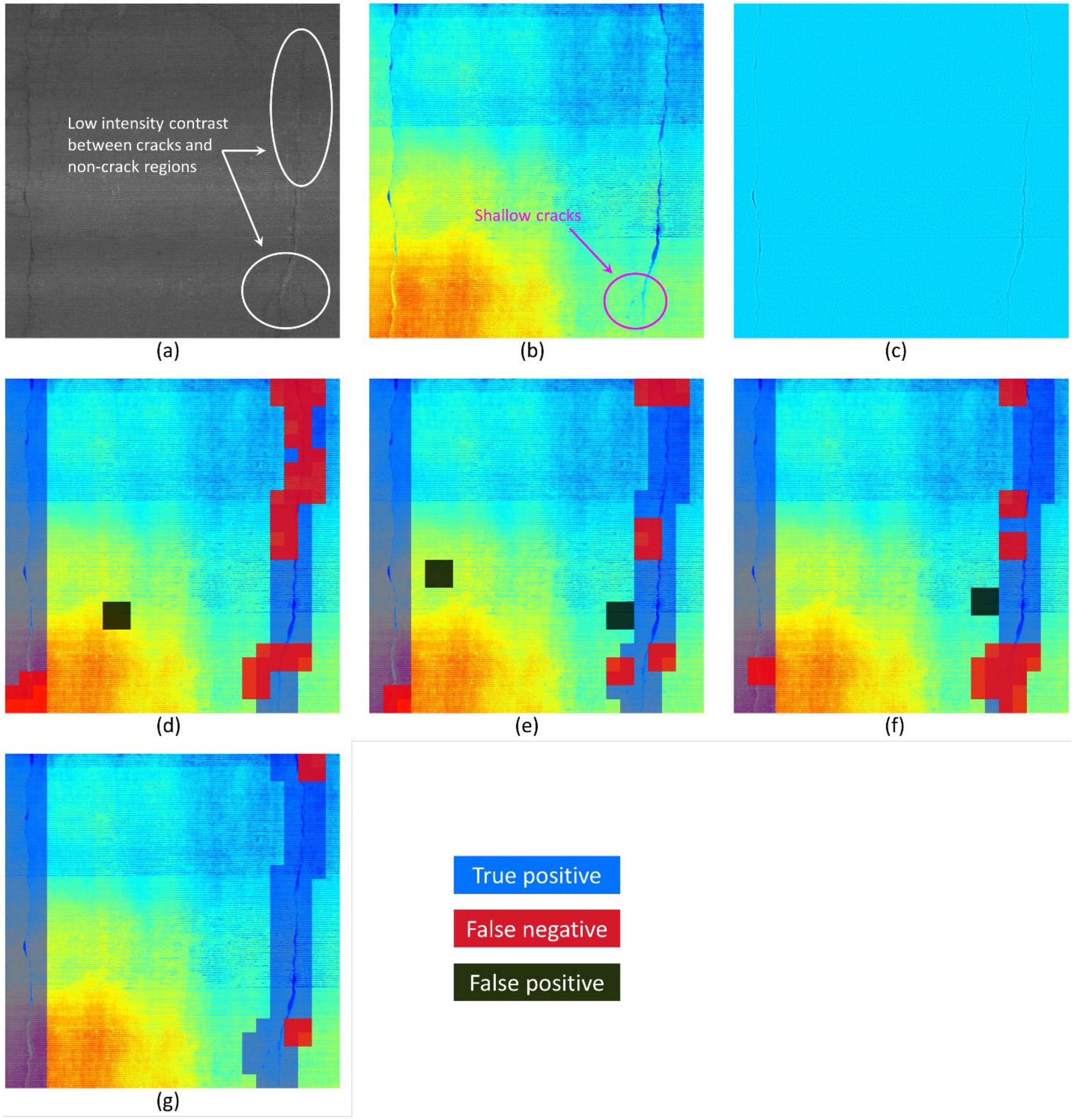


Figure 4-28. Case I: Crack detection on Surface 1: (a) raw intensity image; (b) raw range image; (c) filtered range image; (d) crack map detected from raw intensity image; (e) crack map detected from raw range image; (f) crack map detected from filtered range image; and (g) crack map detected from fused raw image.

- Case II: comparison on network performance by different architectures with data fusion

Through Case I, it is evident that using fused raw image data can achieve the highest classification accuracy. Subsequently, Case II is performed to investigate the better DCNN configuration to exploit the fused raw image data. As described in section 3.4.3, two architectures, Net-B and Net-C, are designed to take fused raw image data for training and testing. The major difference is that Net-B represents a "fuse-extract" pattern while Net-C represents a "extract-fuse" pattern. Case II consists of two subcases (No. 4 and 8 in Table 4-12). The subcases No. 4 and 8 correspond to Net-B and Net-C trained and tested on the fused raw image data, respectively.

The corresponding performance metrics on the test dataset are illustrated in Figure 4-29. Four metrics including Accuracy, Precision, Recall, and F1 are plotted in Figure 4-29 (a). In this plot, the horizontal axis refers to the subcase indexes, which are further grouped according to the performance metrics. It can be seen from Figure 4-29 (a) that Net-B results in higher values in all four metrics by an average margin of around 0.5%; especially, the Recall of Net-B is 0.7% higher than that of Net-C, indicating a lower false-negative rate.

Regarding the network efficiency, Figure 4-29 (b) shows the corresponding training and testing time for both subcases. The horizontal axis of Figure 4-29 (b) is the subcase indexes grouped by the training time and testing time. Net-B demonstrates a higher efficiency than Net-C in training and testing by effectively reducing the time consumption by 7% and 14%, respectively.

Through the above comparison, it can be concluded that Net-B is more accurate and efficient than Net-C on roadway crack classification with data fusion. The "fuse-extract" scheme adopted by Net-B is superior to the "extract-fuse" scheme in Net-C in the sense that it leads to more improvements on the network performance.

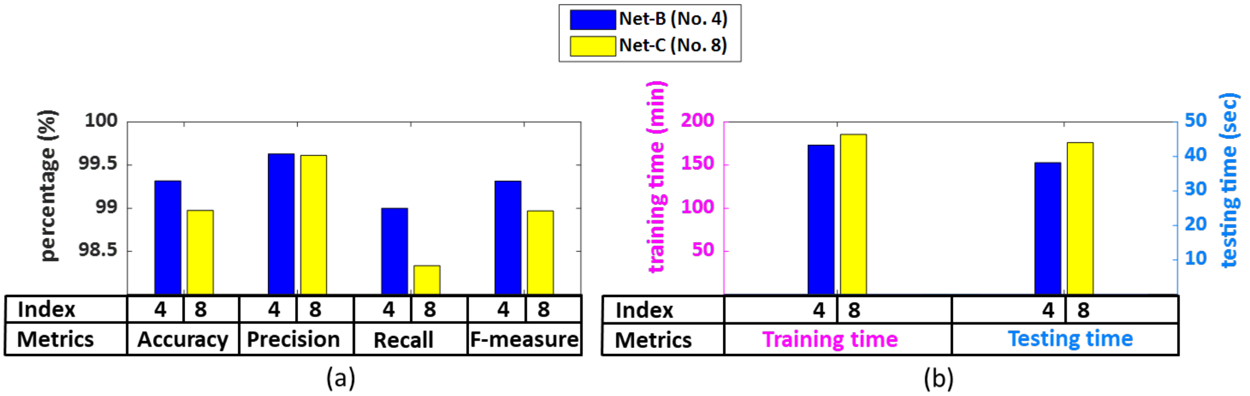


Figure 4-29. Case II: Performance metrics: (a) classification metrics on the test dataset; and (b) training and testing efficiency.

4.4.4 *Limitations*

The proposed methodology based on heterogeneous image fusion, although yielding high classification performance, still have a few limitations. The proposed DCNN methodology is a patch-based approach; that is, the DCNN classifier detects cracks by image patches rather than by pixels, which limits the detection resolution. The proposed methodology may have difficulties detecting cracks on surface regions which suffer from both low intensity contrast in the intensity image and shallow depth in the range image, as shown in Figure 4-28 (g).

4.5 Deep Learning-Based Data Fusion for Crack Segmentation

To disambiguate, the terms “Net-1”, “Net-2”, and “Net-3” in this section refer to the DCNN architectures defined in section 3.4.4.

Section 4.5.1 first introduces the data generation process; then, information on the experimental setup is introduced in section 4.5.2; finally, two experimental cases are presented in section 4.5.3.

4.5.1 *Data Generation*

- Image acquisition and processing

The raw intensity and range image data used in this study is resulted from a long-term (over one year) effort on data collection from multiple concrete roadways, under varying ambient environment and highly complex roadway surface conditions. The acquired roadway images are further cropped into small patches (dimension: 256×256×3) for analysis. In addition to the raw range and intensity data, filtered image data is generated through the filter-based technique [29] developed in section 3.1. Furthermore, fused raw image data is created by combining the raw range and intensity image through the proposed image data fusion approach described in section 3.4.2. In total, over 4000 image patches in the format of raw range, raw intensity, filtered range, and fused raw image data are produced. Figure 4-30 illustrates some examples from the acquired image data with real-world complexities. In Figure 4-30, image-related issues such as surface variations and grooved patterns exist in some range image patches; furthermore, as can be seen, shallow cracks in some range images which are further contaminated by grooved patterns are more distinguishable in the intensity images. Similarly, some intensity image patches suffer from the issue of low contrast between the crack and non-crack regions; however, in their range image counterparts, the cracking features are very noticeable. Besides, in the corresponding filtered range images, the issues of surface variations and grooved patterns are effectively eliminated.

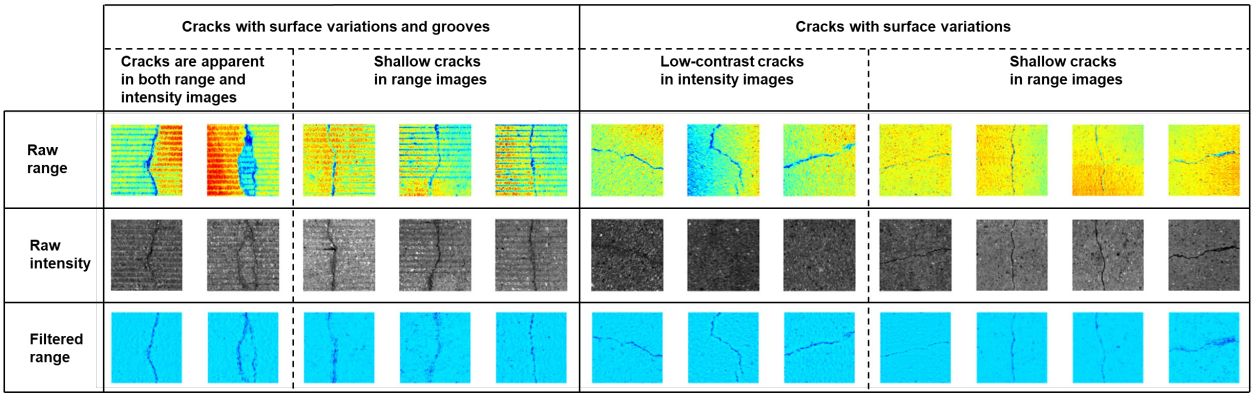

Cracks with surface variations and grooves
Cracks with surface variations
Cracks are apparent in both range and intensity images
Shallow cracks in range images
Low-contrast cracks in intensity images
Shallow cracks in range images
Raw range
Raw intensity
Filtered range

Figure 4-30. Examples of the acquired image data.

- Ground truth generation

The ground truth pixel label map for each image patch is generated through manual labeling by a group of trained personnel over a period of half a year. The manual labeling process is performed by leveraging information from both the range and intensity image data to reduce the uncertainties due to low-contrast cracks in the intensity data or shallow cracks in the range data. It is noted that during training and testing, the four different types of image data share the same ground truth.

- Data augmentation

In this study, data augmentation techniques (section 2.3.3) including rotation (90°, 180°, and 270° counter-clockwise) and mirroring (top-bottom and left-right) are applied to the image patches and their associated ground truth, expanding the total number of image sets to over 25000. Note that each image set consists of a raw range image, a raw intensity image, a filtered range image, a fused raw image, and a ground truth pixel label map.

- Dataset configuration

The generated image data is split into three datasets as tabulated in Table 4-14, namely the training, validation, and test datasets, following a ratio of 60%:20%:20%. The training dataset contains the majority of the image data, which is used during training to help update the learnable parameters of the DCNN. The validation dataset is evaluated every a few iterations during training to help tune the hyperparameters of the DCNN and indicate if overfitting occurs. The test dataset, which does not participate in training, is then evaluated by the trained DCNN models for a cross-comparison on their DCNN performance. Table 4-14 shows the detailed configuration of the generated datasets. It is noted in Table 4-14 that the same configuration (i.e., image sequence) on the training, validation, and testing datasets is applied to different types of image data. Thus, by training and testing Net-1, 2 and 3 on the image data with distinct characteristics, the impact from heterogeneous image data can be revealed and compared.

4.5.2 *Experimental Setup*

- Computing hardware and software

The specifications of the data processing computer are as follows: CPU: Intel i7-8750H; GPU: Nvidia GTX 1060 with 6GB RAM. The proposed methodology is implemented in MATLAB R2019a with its deep learning toolbox [143].

- Hyperparameter configuration

This study adopts the mini-batch SGD with momentum algorithm as introduced in section 2.3.4 to train the DCNNs. Several hyperparameters including the weight decay factor, momentum, learning rate, LReLU factor, mini-batch size, number of epochs are involved in the optimization process. Upon training different architectures, the same hyperparameter values are adopted to provide a basis for comparison. The hyperparameter values are as follows: weight decay factor = 0.0003; momentum = 0.9; initial learning rate = 0.01; learning rate drop factor = 0.8; LReLU factor = 0.01; mini-batch size = 10; number of epochs = 10. Besides, the validation dataset is evaluated every half an epoch during training.

- Parameter initialization

The learnable parameters in the proposed DCNNs are initialized base on the settings in section 2.3.4.

4.5.3 *Results and Discussions*

In this section, two experimental cases are performed. In Case I, eight DCNNs with different image data are trained and tested for comparison. In Case II, the segmentation performance of these DCNNs is further demonstrated on a concrete roadway surface.

- Case I: Impacts from heterogeneous image data

In addition to the proposed DCNNs in this study, two benchmark architectures, namely the CrackNet II [30] and VGG16-FCN [83] are employed for comparison. CrackNet II is designed to analyze laser-scanned range images, with a surface flattening technique applied as pre-processing to address the issue of surface variations [30]; VGG16-FCN is designed to take raw intensity image. Case I consists of eight subcases (see Table 4-15), which are specified as follows: *i*) Net-1 with raw range image (No. 1 in Table 4-15, same hereafter); *ii*) Net-1 with raw intensity image (No. 2); *iii*) Net-1 with filtered range image (No. 3); *iv*) Net-2 with fused raw image (No. 4); *v*) Net-3 with fused raw image (No. 5); *vi*) CrackNet II with raw range image (No. 6); *vii*) CrackNet II with filtered range image (No. 7); and *viii*) VGG16-FCN with raw intensity image (No. 8).

Table 4-14. Case I: Configuration of the image datasets.

Architecture	Data type	Number of crack images and ground truth			
		Training dataset	Validation dataset	Testing dataset	**Total**
Net-1 /VGG16-FCN	raw intensity	15016	5005	5005	25026
Net-1 /CrackNet II	raw range	15016	5005	5005	25026
Net-1 /CrackNet II	filtered range	15016	5005	5005	25026
Net-2 /Net-3	fused raw image	15016	5005	5005	25026

Table 4-15. Case I: Performance metrics.

Index	Architecture	Data type	Validation					Testing					Time	
			Precision (%)	Recall (%)	F1 (%)	IOU (%)	BF score (%)	Precision (%)	Recall (%)	F1 (%)	IOU (%)	BF score (%)	Training (min)	Testing (sec)
1	Net-1	raw range	84.0	71.3	75.6	61.9	79.5	84.0	71.3	75.6	61.8	79.4	959.5	241.4
2	Net-1	raw intensity	76.2	58.1	63.3	50.8	68.5	76.2	57.7	63.0	50.4	68.2	977.5	246.2
3	Net-1	filtered range	87.0	68.4	75.1	61.4	79.1	87.0	68.7	75.3	61.6	79.2	977.8	286.7
4	Net-2	fused raw image	87.5	76.1	80.3	68.3	84.4	87.5	76.2	80.3	68.3	84.4	1013.9	313.9
5	Net-3	fused raw image	88.0	**76.7**	**80.8**	**69.0**	**84.9**	88.0	**76.6**	**80.7**	**68.9**	**84.9**	1093.9	350.4
6	CrackNet II	raw range	**92.6**	24.7	36.1	24.0	39.8	**92.5**	24.7	36.0	23.9	39.7	2404.1	326.8
7	CrackNet II	filtered range	90.9	41.8	53.2	39.3	57.3	90.8	42.4	53.7	39.7	57.7	2445.0	404.5
8	VGG16-FCN	raw intensity	74.1	59.4	63.4	50.6	67.3	73.4	58.7	62.8	50.0	66.8	1199.4	351.5

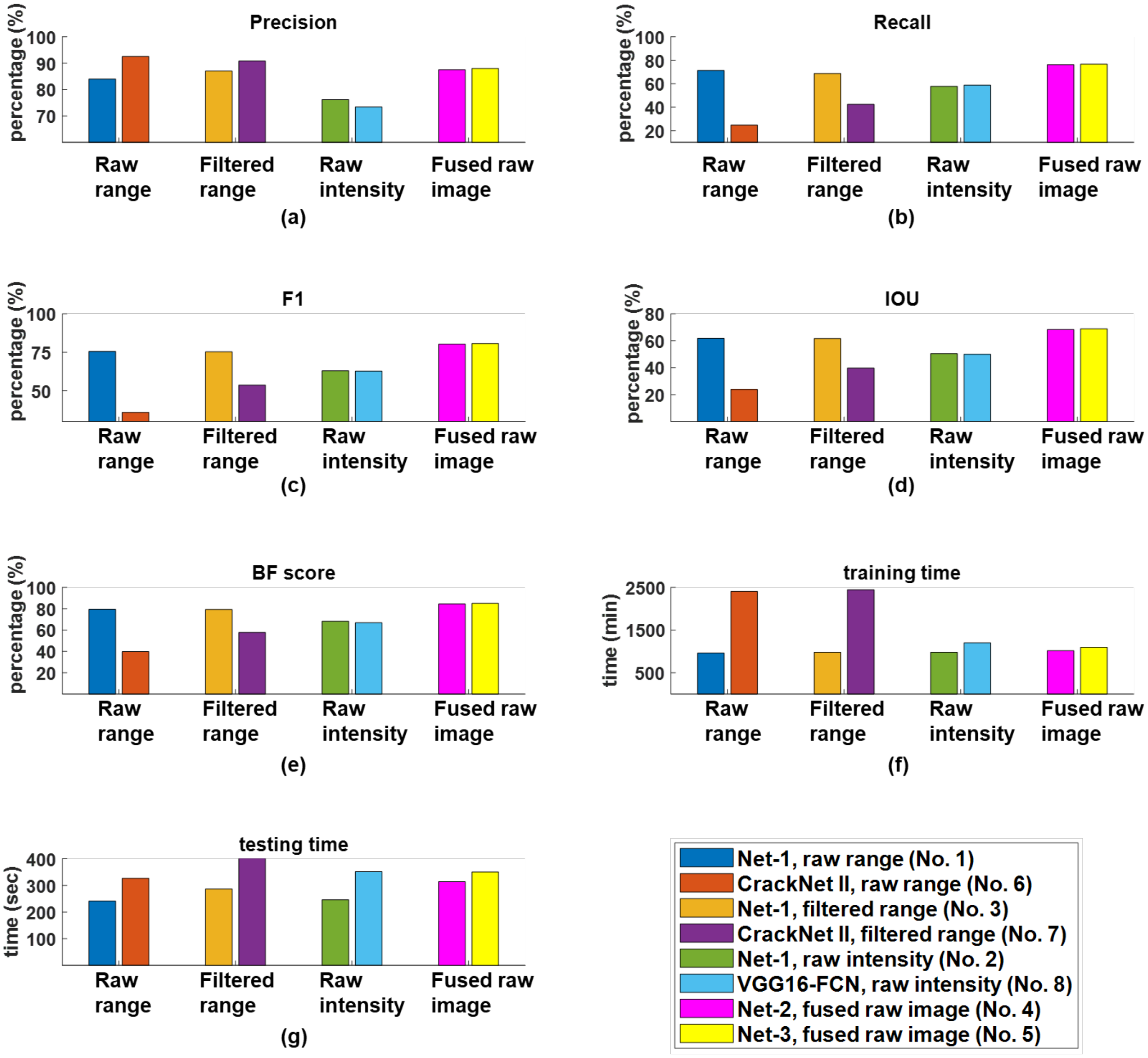


Figure 4-31. Case I: Bar plots of the testing performance metrics: (a) average Precision; (b) average Recall; (c) average F1; (d) average IOU; (e) average BF score; (f) training time; and (g) testing time.

The detailed performance statistics of the eight subcases are tabulated in Table 4-15 with the bold font faced values indicating the best performance in each metric. Meanwhile, metrics including the average Precision, Recall, F1 score, IOU, and BF score, which are evaluated on the test dataset, are illustrated in Figure 4-31 (a-e). Besides, the training time and testing time by each DCNN are also illustrated in Figure 4-31 (f) and (g), respectively. In Figure 4-31, the horizontal axis of each plot refers to the four different types of image data: raw range, filtered range, raw intensity, and fused raw image. Several observations can be made from Figure 4-31 and Table 4-15:

i) Comparison between the subcases No. 1 through 4: It can be observed as a general trend that Net-2 with fused raw image (No. 4) yields much higher segmentation accuracy than Net-1 with raw range (No. 1), raw intensity (No. 2), and filtered range image data (No. 3). Meanwhile, using raw intensity image leads to the worst segmentation performance, due to the existence of low-contrast cracks and other image disturbances in the intensity image data. Taking the statistics of the average F1 score as an example, by using Net-2 with the fused raw image data (No. 4), the increase in the F1 score is 4.9% (vs. Net-1 with raw range image), 18.5% (vs. Net-1 with raw intensity image), and 7.5% (vs. Net-1 with filtered range image), respectively. Considering that Net-1 and Net-2 share the same architecture layout except the image input layer and the first convolutional layer, such a significant improvement on the segmentation accuracy is attributed to the proposed data fusion strategy. Also, despite the volume of image data is doubled during analysis, applying data fusion does not result in a deteriorated model efficiency. The training and testing time by Net-2 with fused raw image (No. 4) are only increased by 4.4% and 21.6%, respectively, compared to the average time consumption by the other three subcases (No. 1, 2, and 3);

ii) Comparison between No. 4 and No.5: From Table 4-15, given the same type of image data used (i.e., fused raw image), Net-3 (No. 5) has very similar performance as Net-2 (No. 4), where Net-3 basically outperforms Net-2 by an increase of around 0.5% in all performance metrics. However, Net-3 does require 7.9% and 11.4% more time than Net-2 on training and testing, respectively. Note that Net-2 and Net-3 represent the “fuse-extract” and “extract-fuse” patterns, with respect to the means to exploit the fused raw image. Judging from the segmentation accuracy, it can be concluded that Net-3, which employs the “extract-fuse” pattern, leads to a slightly better performance;

iii) Comparison between the proposed DCNNs (No. 1, 3, 2) and the benchmarks (No. 6, 7, 8): For the subcases No. 1 and 6 both utilizing the raw range image, it can be seen that Net-1 (No. 1) almost exceeds CrackNet II in every performance metric except the average Precision value. The reason CrackNet II has very poor performance may be that it requires to apply a pre-processing technique (i.e., surface flattening) to address the issue of surface variation [30] prior to DCNN analysis. By comparing Net-1 (No. 3) against CrackNet II (No. 7), which are both trained and tested on the filtered range image, Net-1 leads to higher values in the average Recall, F1, IOU, and BF score on the test dataset, with an improvement of 26.3%, 21.6%, 21.9%, and 21.5%, respectively. Still, the average Precision value by CrackNet II is 3.8% higher than by Net-2. Regarding the DCNNs using intensity image data, Net-1 (No. 2) performs slightly better than VGG16-FCN (No. 8), as measured by their metrics values. Net-1 results in higher values in the average Precision, F1, IOU, and BF score with an improvement of 2.8%, 0.2%, 0.4%, and 1.4%, respectively, while VGG16-FCN has a 1.0% higher average Recall value. Regarding the model efficiency, the proposed DCNNs consume less time on both training and testing; for example, Net-1 with filtered range image (No. 3) only consumes about 40.0% and 70.9% of the training and testing time by CrackNet II (No. 7), respectively. It is thus demonstrated that the proposed DCNNs are capable to achieve the same level or even better segmentation performance and efficiency compared to the benchmarks;

iv) Comparison regarding the effects of image pre-processing: Net-1 is trained and tested by the raw range image (No. 1) or filtered range image data (No. 3). Note that the raw range image data contains image disturbances including surface variations, grooved patterns, and scanning noises, while the filtered range images are free from such disturbances. However, the differences between the subcases No. 1 and 3 in the testing metrics are very marginal, which are only -3.0%, 2.6%, 0.3%, 0.2%, and 0.2% for the average Precision, Recall, F1, IOU, and BF score, respectively. The robustness of the proposed DCNN is shown in that it does not require any image pre-processing procedures to address the image-related issues, and that its segmentation performance does not deteriorate under the influence of the image disturbances in the raw range image data. Another observation made on CrackNet II (No. 6 vs. 7) is that the image pre-processing procedure does help improve the segmentation performance by CrackNet II, because it requires a procedure to address the issue of surface variation in the range image data prior to DCNN analysis.

In addition to comparing the average values by the bar plots in Figure 4-31, the histograms of five performance metrics, evaluated on the test dataset which contains 5005 images, are also provided in Figure 4-32 and Figure 4-33. In the histogram plot, the horizontal axis is the percentage value of the metric, and the vertical axis refers to the frequency counts. And, each column in both figures represents a type of the performance metrics, which include the Precision, Recall, F1, IOU, and BF score. Figure 4-32 focuses on comparing the proposed DCNNs with different types of image data. Again, similar conclusions regarding the impact from heterogeneous image data can be drawn. In Figure 4-32 (a), (b), and (c), Net-2 with fused raw image is compared against Net-1 with raw range [Figure 4-32 (a)], raw intensity [Figure 4-32 (b)], and filtered range image [Figure 4-32 (c)]. Among the four subcases, Net-2 with fused raw image leads to the best segmentation performance, indicated by the higher mean and less spread of its histograms. This result again demonstrates the effectiveness of data fusion through cross-domain feature correlation to address the uncertainties in the data. Meanwhile, Net-2 and Net-3, both of which exploit the fused raw image data, yield very similar segmentation accuracy, as their histograms have a high degree of overlap in Figure 4-32 (d). Finally, by comparing the histograms by the subcases No. 1 (Net-1 with raw range image) and No. 3 (Net-1 with filtered range image), a conclusion can be drawn that the image pre-processing does not yield any significant improvement on the segmentation accuracy by Net-1, because their histograms in Figure 4-32 (e) are highly overlapped with each other.

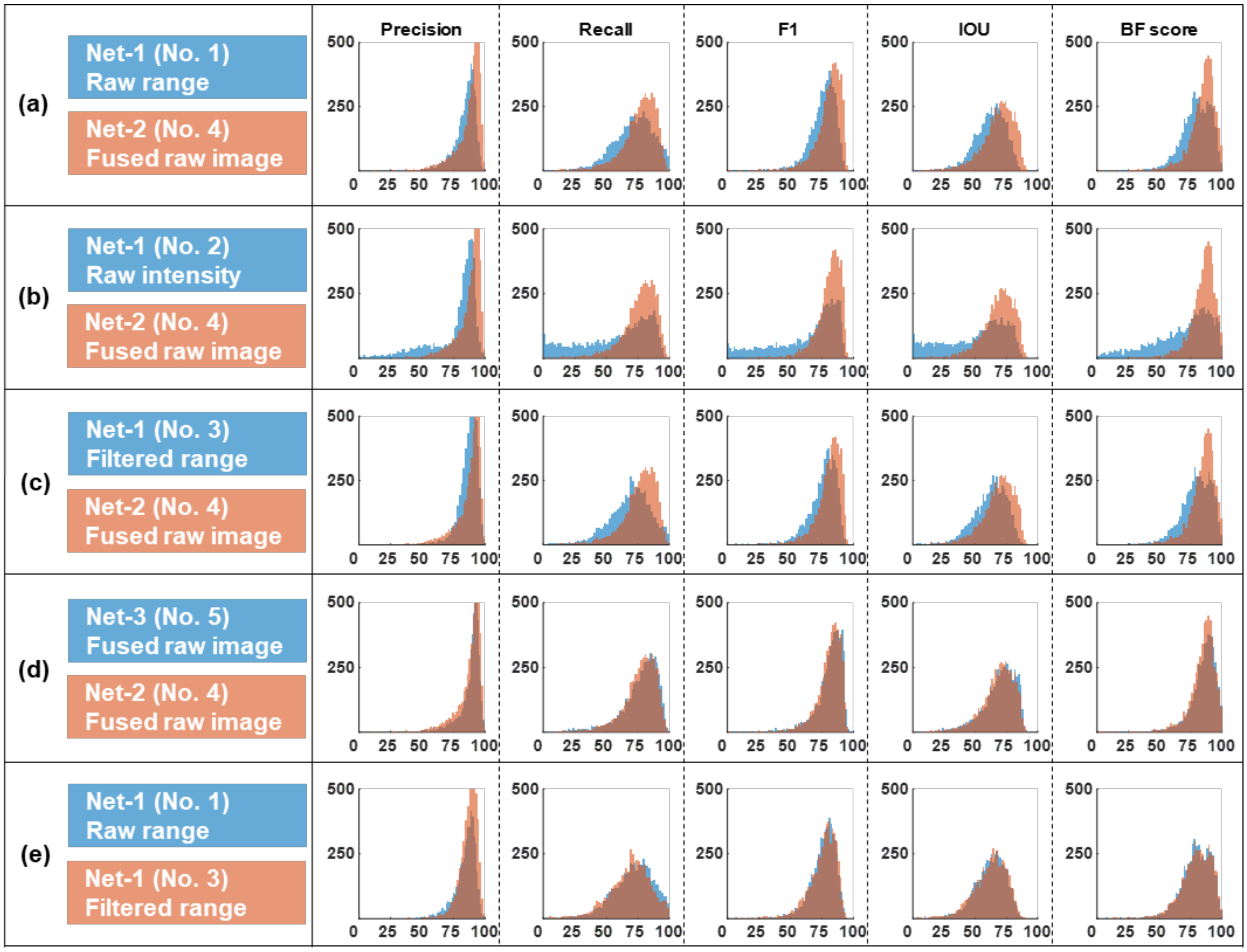


Figure 4-32. Case I: Histograms of the testing metrics by the proposed DCNNs: (a) Net-1 (raw range) vs. Net-2 (fused raw image); (b) Net-1 (raw intensity) vs. Net-2 (fused raw image); (c) Net-1 (filtered range) vs. Net-2 (fused raw range); (d) Net-3 (fused raw range) vs. Net-2 (fused raw range); and (e) Net-1 (raw range) vs. Net-1 (filtered range).

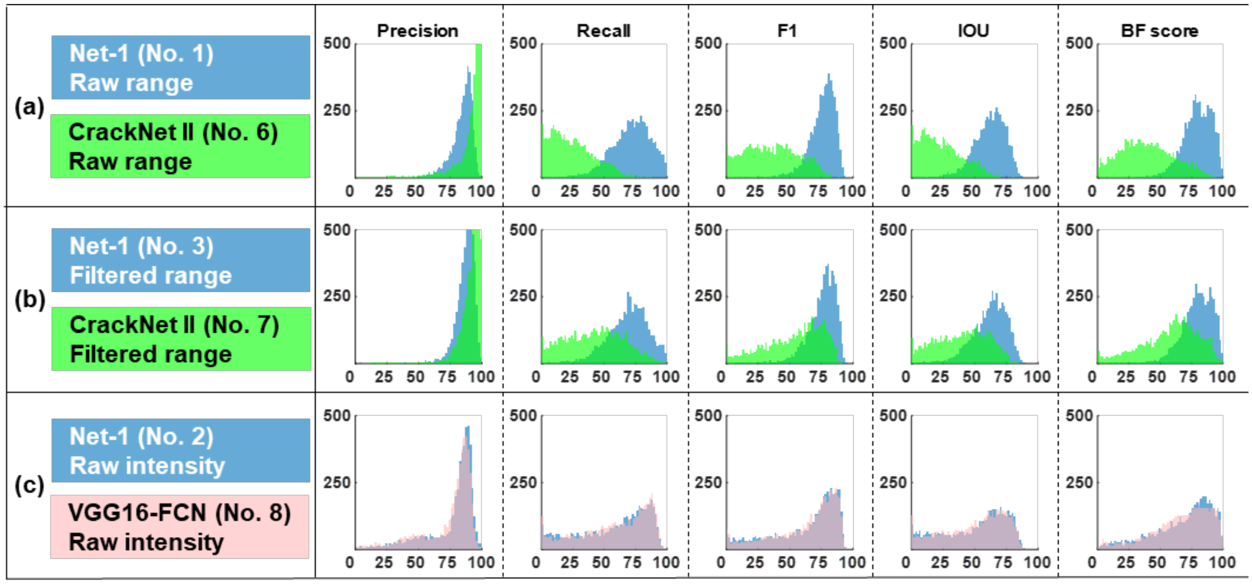


Figure 4-33. Case I: Histograms of the testing metrices by the proposed DCNNs and benchmarks: (a) Net-1 vs. CrackNet II with raw range image; (b) Net-1 vs. CrackNet II with filtered range image; and (c) Net-1 vs. VGG16-FCN with raw intensity image.

Figure 4-33 illustrates the histograms by the proposed DCNNs and the benchmarks. As shown by Figure 4-33 (a) and (b), when using the raw range or filtered range image for analysis, Net-1 outperforms CrackNet II by a significant level. The histograms of the Recall, F1, IOU, and BF score by Net-1 are more concentrated to the right end of each plot, indicating more accurate and consistent segmentation performance. Nevertheless, CrackNet II has better performance in terms of the Precision value. Figure 4-33 (c) shows the comparison between Net-1 (No. 2) and VGG16-FCN (No. 8), where their histograms overlay with each other. However, in general, the performance by both Net-1 and VGG16-FCN with raw intensity image are similar and quite poor; their histograms have a very large dispersion, indicating that a large proportion of the predicted pixel label maps have very low metrics values.

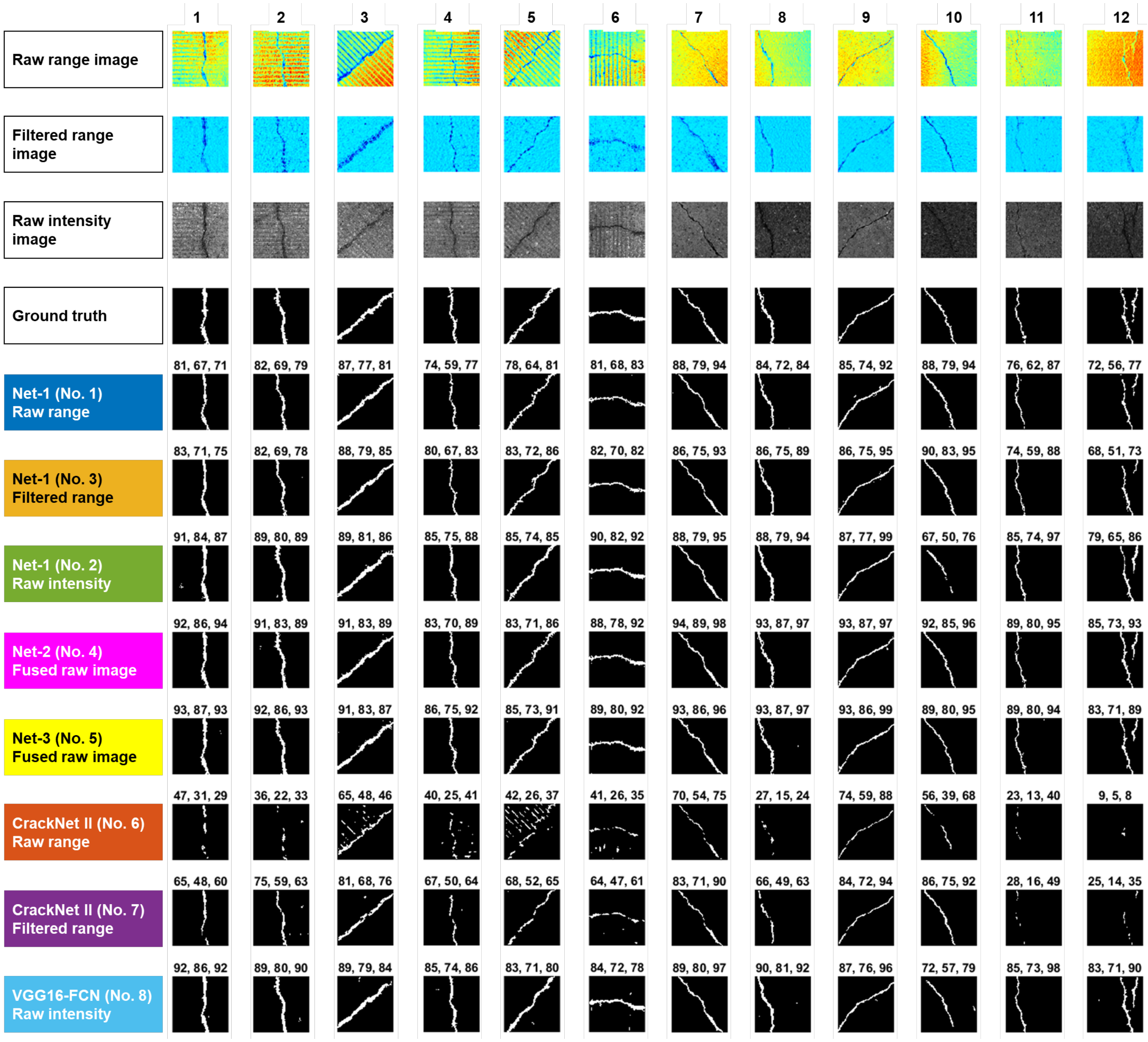


Figure 4-34. Case I: examples of crack segmentation on image patches.

To further demonstrate the DCNN segmentation performance, prediction results on twelve image patches with diverse data characteristics are illustrated in Figure 4-34. From the top to bottom of Figure 4-34, image data including the raw range, filtered range, raw intensity image, and ground truth pixel label map are displayed; then, the prediction results by different DCNNs are illustrated sequentially. Meanwhile, the percentage values of three metrics including the F1, IOU, and BF score are labelled on top of each predicted pixel label map. It can be seen in Figure 4-34 that the image patches to be demonstrated have very diverse characteristics, such as surface variations, grooved patterns, shallow cracks in range images, and low-contrast cracks in intensity images. Judging from the metrics values, Net-2 with fused raw image data (No. 4) yields better performance on almost all image patches than Net-1 with raw range (No. 1), filtered range (No. 3), and raw intensity image data (No. 2), demonstrating the effectiveness of the proposed data fusion strategy. Meanwhile, Net-2 (No. 4) and Net-3 (No. 5), which utilize the fused raw image data, have very similar segmentation performance. It is demonstrated that by leveraging the spatial co-registration feature in the fused raw image data, the uncertainties (e.g., shallow cracks in range images and low-contrast cracks in intensity images) can be alleviated through cross-domain feature correlation. Regarding the benchmarks, the segmentation performance of CrackNet II with either raw range or filtered range image data is not as good as the proposed DCNNs, especially when grooved patterns (patches 3, 4, 5, 6 by No. 6) or shallow cracks (patches 11, 12 by No. 6 and 7) exist. VGG16-FCN (No. 8) has very consistent performance on the intensity images, which yields similar metrics values as the proposed Net-1 with intensity images (No. 2).

In addition, the effect of heterogeneous image fusion on the segmentation performance of the proposed DCNNs (Net-1 and 2) is demonstrated in Figure 4-35, where another twelve image patches suffering from the issues of shallow cracks in range images or/and low intensity contrast in intensity images are tested by the proposed DCNNs. In Figure 4-35, the raw range image, raw intensity image, ground truth, and prediction results (by Net-1 with raw range image, Net-1 with raw intensity image, and Net-2 with fused raw image) are displayed from top to bottom. Again, three metrics including F1, IOU, and BF score are labelled on top of each predicted pixel label map. As can be seen in Figure 4-35 (a), when shallow cracks exist in the range images, Net-1 (No. 1) which only relies on raw range image is not able to detect the cracks with shallow depths; and, vice versa for the intensity image cases in Figure 4-35 (b), where the low intensity contrast between cracks and non-crack regions causes false negative detections by Net-1 with raw intensity image. However, the proposed Net-2 which utilizes the fused raw image data can alleviate such image disturbances existing in the individual data sources [Figure 4-35 (a) and (b)] and reduce false negative detections through cross-domain feature correlation, leading to the highest segmentation performance. Furthermore, the four image patches in Figure 4-35 (c) demonstrate a mixture of shallow cracks in range images and low intensity contrast in intensity images. However, these two types of issues do not occur at the same location. Thus, the information in the range and intensity images complements each other, which has been leveraged by Net-2 when using the fused raw images. More specifically, due to the image-related issues, Net-1 using either type of image data alone yields false negative detections. By leveraging the fused raw image data, Net-2 can reduce such uncertainties through cross-domain feature correlation, as indicated by the fact that the prediction results by Net-2 yield the highest metrics values, and that the false negative detections are effectively reduced as shown in the figure.

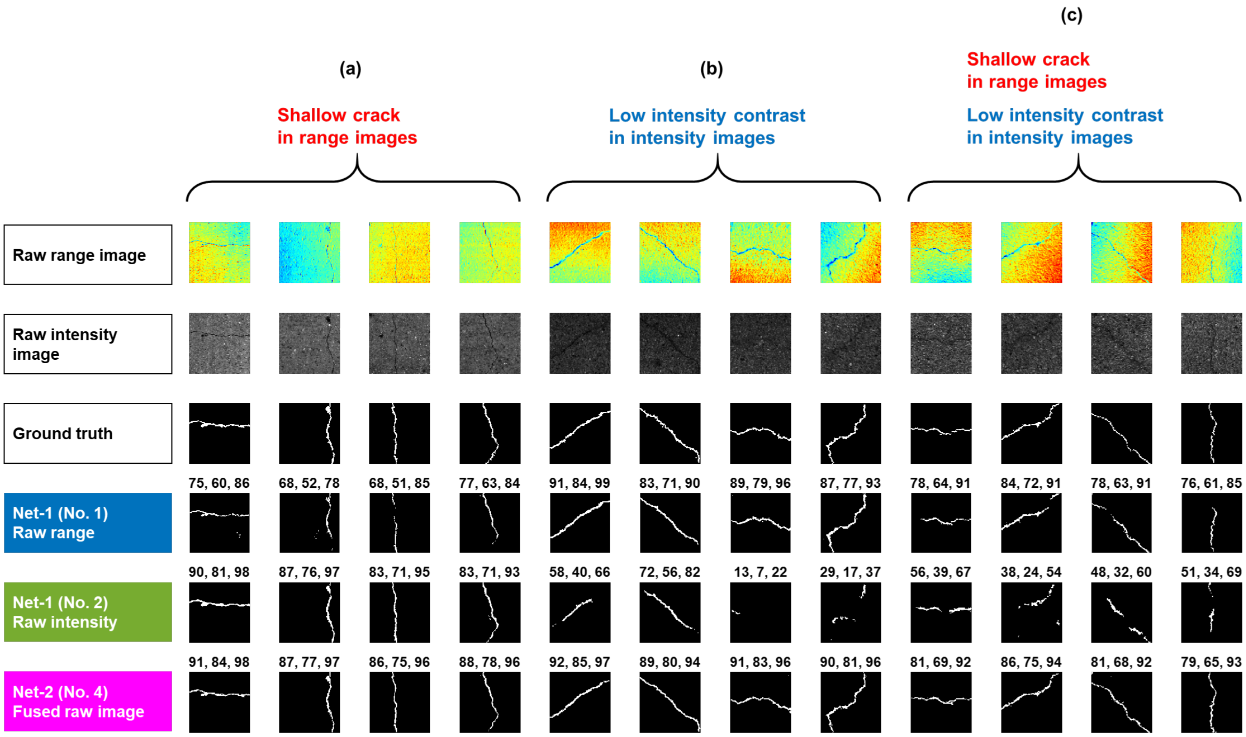


Figure 4-35. Case I: examples of data fusion to improve segmentation performance through cross-domain feature correlation.

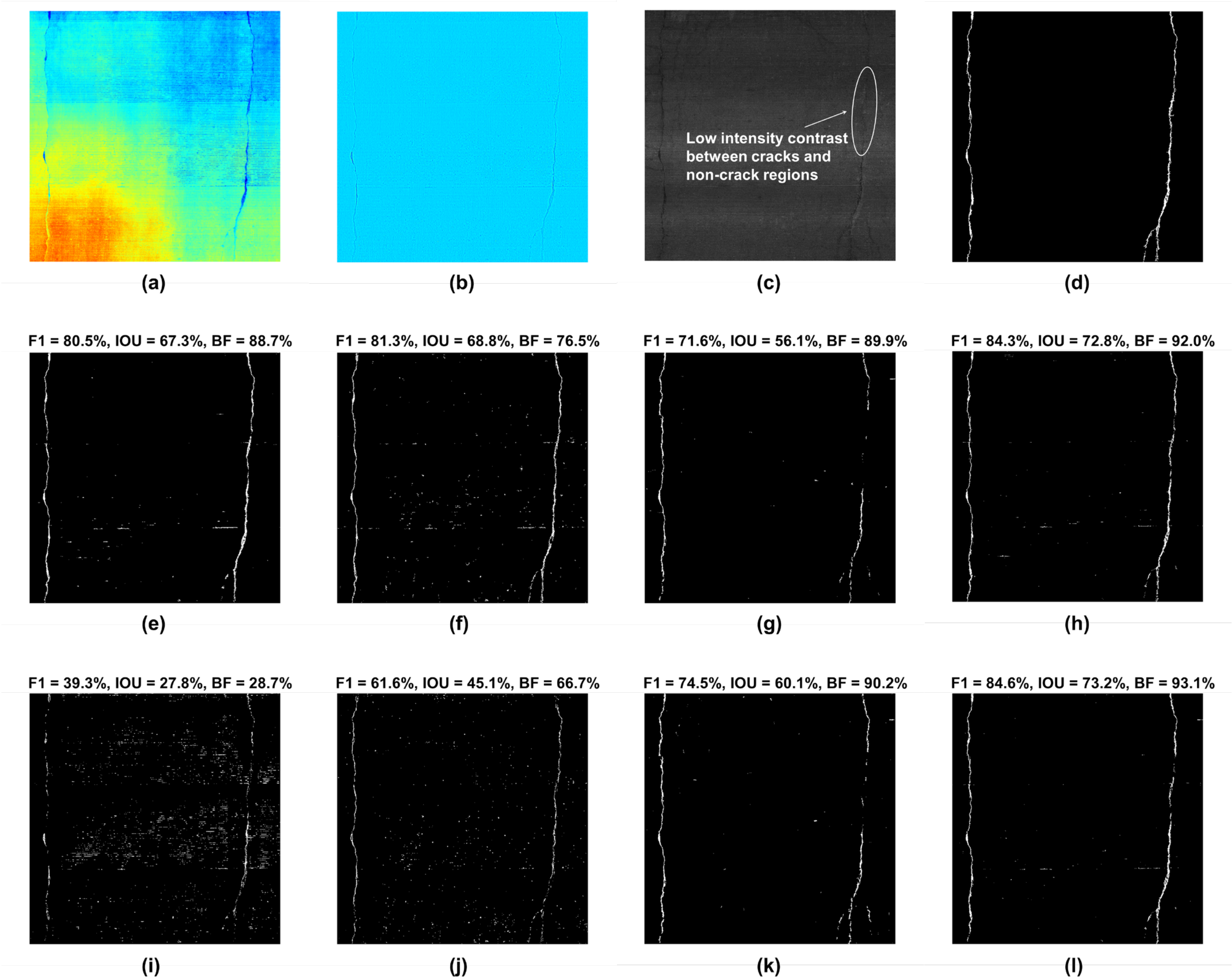

Low intensity contrast between cracks and non-crack regions
(a)
(b)
(c)
(d)
F1 = 80.5%, IOU = 67.3%, BF = 88.7%
F1 = 81.3%, IOU = 68.8%, BF = 76.5%
F1 = 71.6%, IOU = 56.1%, BF = 89.9%
F1 = 84.3%, IOU = 72.8%, BF = 92.0%
(e)
(f)
(g)
(h)
F1 = 39.3%, IOU = 27.8%, BF = 28.7%
F1 = 61.6%, IOU = 45.1%, BF = 66.7%
F1 = 74.5%, IOU = 60.1%, BF = 90.2%
F1 = 84.6%, IOU = 73.2%, BF = 93.1%
(i)
(j)
(k)
(l)

Figure 4-36. Case II: Segmentation performance on a concrete roadway surface: (a) raw range image; (b) filtered range image; (c) raw intensity image; (d) ground truth; (e) predicted by Net-1 with raw range image; (f) Net-1 with filtered range image; (g) Net-1 with raw intensity image; (h) Net-2 with fused raw image; (i) CrackNet II with raw range image; (j) CrackNet II with filtered range image; (k) VGG16-FCN with raw intensity image; and (l) Net-3 with fused raw image.

- Case II: Demonstration of DCNN performance on a concrete roadway surface

After the DCNNs are trained and tested on image patches in Case I, Case II further demonstrates their segmentation performance with real-world complexity by applying them on an entire concrete roadway surface (3072×3072 pixels), representing a single lane of 6-meter long roadway. Note that this roadway data has not participated in the training in Case I. Figure 4-36 (a-d) illustrate the raw range, filtered range, raw intensity image, and ground truth label map of the roadway surface, respectively. This roadway surface suffers from issues such as grooved patterns and surface variations in the range image [see Figure 4-36 (a)], and low contrast between cracks and non-crack regions in the intensity image as indicated by the circles in Figure 4-36 (c). Prediction results by the proposed DCNNs and the benchmarks are displayed in Figure 4-36 (e-l), respectively, with three performance metrics including the F1, IOU, and BF score labelled on top of each predicted pixel label map. Meanwhile, the detailed statistics of these metrics together with the testing time are tabulated in Table 4-16.

Table 4-16. Case II: Performance metrics.

Index	Architecture	Data type	Testing metrics			Testing time (sec)
			F1 (%)	IOU (%)	BF score (%)	
1	Net-1	raw range	80.5	67.3	88.7	31.1
2	Net-1	raw intensity	71.6	56.1	89.9	28.3
3	Net-1	filtered range	81.3	68.8	76.5	28.9
4	Net-2	fused raw image	84.3	72.8	92.0	31.4
5	Net-3	fused raw image	**84.6**	**73.2**	**93.1**	38.3
6	CrackNet II	raw range	39.3	27.8	28.7	68.3
7	CrackNet II	filtered range	61.6	45.1	66.7	68.6
8	VGG16-FCN	raw intensity	74.5	60.1	90.2	40.3

When using the raw range image for analysis, Net-1 [Figure 4-36 (e)] shows good segmentation performance, with the F1, IOU, and BF score as 80.5%, 67.3%, and 88.7%, respectively. However, for the subcase of CrackNet II with raw range image, many false detections as shown in Figure 4-36 (i) are created due to the issues of grooved patterns and surface variations. Once the filtered range image is utilized, in which the image-related issues are suppressed, CrackNet II [Figure 4-36 (j)] yields better performance than before [Figure 4-36 (i)]. Meanwhile, judging from the metrics, the segmentation performance by Net-1 with filtered range [Figure 4-36 (f)] vs. raw range image [Figure 4-36 (e)] is quite similar. In Figure 4-36 (g) and (k), where the prediction results by Net-1 and VGG16-FCN with raw intensity image are presented, it can be observed that both DCNNs experience a significant deterioration in the segmentation performance due to the issue of low intensity contrast between cracks and non-crack regions. Regarding the DCNN efficiency, as shown in Table 4-16, the proposed DCNNs are more efficient than the benchmarks, where the testing time by the proposed DCNNs is only up to 70% of by the benchmarks (e.g., Net-1 vs. VGG16-FCN with raw intensity image).

Among all eight subcases in Case II, Net-2 [Figure 4-36 (h)] and 3 [Figure 4-36 (l)], which exploit the fused raw image data, lead to much better performance than the other DCNNs relying on a single type of image data. Through a comparison between Figure 4-36 (g) and (h), it is shown that, using the fused raw image data in Net-2 can address the issue of low contrast in the intensity image, by referring to the pixel locations in the corresponding raw range image where crack patterns exist. Accordingly, the segmentation performance by using the fused raw image is much better than using the raw intensity data alone. Furthermore, judging from the performance metrics by Net-2 and 3, which both exploit the fused raw image data, Net-3 exceeds Net-2 by a marginal difference as follows: 0.3%, 0.4%, and 1.1% in the F1, IOU, and BF score, respectively. Nevertheless, as shown in Table 4-16, Net-2 yields less testing time than Net-3. It is thus demonstrated through both Case I and II that, Net-3, representing the "extract-fuse" pattern, yields slightly better segmentation performance than Net-2 which represents the "fuse-extract" pattern.

4.5.4 *Limitations*

Few limitations of this proposed image fusion methodology include: *i*) pixel-to-pixel spatial correspondence between the range and intensity image data is a prerequisite of the proposed data fusion process; *ii*) this methodology may have difficulties detecting cracks in image data which suffers from both low contrast in the intensity image and shallow depths in the range image.

CHAPTER 5: DISCUSSION AND FUTURE WORK

In this chapter, first, summaries and conclusions are drawn from the experimental study section for each proposed methodology; then, an overall evaluation and conclusion on this technical report is provided.

5.1 Image Processing Technique for Robust Crack Detection Using Range Image Data

In section 3.1, a robust crack detection methodology utilizing frequency domain filtering and contouring analysis on range image data is proposed. The proposed methodology relies on elevation difference to detect cracks, assuming the cracks are lower than the surroundings in local image regions. The use of range image instead of intensity image reduces the influence from varying illumination condition, blemishes, and low contrast between crack pixels and their surrounding pixels.

Frequency domain filtering techniques are applied as image pre-processing to address the following issues: surface variations, image noises, and grooved patterns. Unlike traditional pre-processing methods, which typically require subjective parameters, the frequency domain filtering developed in this methodology is designed based on a physical relationship between the cutoff frequency and crack width. The experimental results show that it can provide robust and consistent results in practice and is independent from images or operating personnel.

After image pre-processing, a crack detection methodology based on contouring analysis is developed to extract the cracks from the filtered range surface. In the experimental study (section 4.1), crack detection on three bridge deck surfaces utilizing the proposed methodology is presented. The range surfaces vary drastically in noise level, average elevation, and crack pattern, etc. A validation study using the precision-recall analysis is employed to examine the accuracy of the proposed methodology. High metrics values are consistently achieved from the crack detection examples on three range images, indicating the effectiveness and accuracy of the proposed methodology on surface crack extraction.

Through this filter-based methodology, Challenges 2, 3, and 4 as described in section 1.2 are effectively addressed. Nevertheless, because this methodology belongs to the category of non-learning-based methodology, the uncertainties due to the image pre-processing procedure cannot be completely eliminated, despite the robust performance of this methodology as demonstrated through the experimental study. It is thus preferable to develop learning-based crack detection methodologies for better performance and wider applicability. Besides, it is difficult for this methodology to detect shallow cracks in range image data.

5.2 Deep Learning-Based Crack Classification

By offering an optimal DCNN architecture that utilizes raw range images, the methodology proposed in section 3.2 tackled two issues that have not yet been thoroughly addressed in DCNN-based crack classification research. The first issue lies in the disturbances in laser-scanned range image data, including surface variations and non-crack patterns, which demands a crack classification tool with robust performance. The second issue is related with the hyperparameter selection upon using DCNN for roadway crack classification on laser-scanned range images. A comprehensive comparative study is performed to address these issues.

In the study presented in section 4.2, four cases are performed:

i) Case I investigates the optimal hyperparameter values related with network structure, which include network depth, kernel sizes, and network width. The 7-layer architecture with 7×7 convolutional kernels and ascending network widths yields the optimal performance among the proposed DCNN candidates, which may be that it can best describe and reflect the real-world complexity of the acquired laser-scanned range image data;

ii) Case II is focused on the selection of the hyperparameters that are related with training, which include the mini-batch size, initial learning rate and learning rate drop factor, dropout factor, and LReLU factor. The experimental results show that the optimal values of these hyperparameters for roadway range image data are as follows: mini-batch size = 10; initial learning rate = 0.01; learning rate drop factor = 0.8; dropout factor = 50%; LReLU factor = 0.01;

iii) Case III validates the performance of the proposed architecture on crack classification through a comparison with four benchmark architectures. Consistent results on the F1 scores of two test datasets of diverse image data are observed, showing the proposed architecture can achieve a similar level of classification accuracy as the benchmarks but with a much higher efficiency;

iv) In Case IV, the performance and robustness of the proposed DCNN classifier is further demonstrated on a new dataset from actual roadway surveys, which contains three cracked image surfaces contaminated with non-crack patterns. The predicted crack map with a high testing F1 score demonstrates the capability of the proposed methodology on detecting cracks under the disturbances of many non-crack patterns such as grooves, pavement edges, and shoulder drop-offs.

The hyperparameter selection process developed in this study and the conclusions summarized from the experiments can provide insights to similar civil engineering applications using laser-scanned range images for roadway crack classification. This proposed methodology helps address Challenge 5 (section 1.2) regarding the optimal hyperparameter configuration for DCNN-based crack classification applications; moreover, it tackles Challenges 2 and 3 by directly takes raw range images as the input without any pre-processing procedures. Still, this methodology may have difficulties detecting cracks with shallow depths as exposed in section 4.2.4.

5.3 Deep Learning-Based Crack Segmentation

The proposed methodology in section 3.3 is focused on addressing the issues of surface variations and grooved patterns in range image data, which are very challenging to current DCNN-based roadway crack segmentation applications using range images.

In the experimental study (section 4.3), two experimental cases are performed. Case I is designed to investigate the influences from different architecture layouts on the segmentation performance. The following concluding remarks from Case I are summarized:

i) Based on the results of the proposed DCNN architectures, the use of residual connections yields higher metrics values, and it prevents performance degradation especially in deeper architectures;

ii) From the illustrative results in Figure 4-20, it is shown that the proposed Net-2 through Net-6 can achieve high segmentation performance under the disturbance of surface variations and grooves due to their deep architecture layouts with residual connections;

iii) Among the proposed DCNN architectures, Net-4 leads to the highest values in all the performance metrics on the test dataset, thus it is determined as the optimal architecture for roadway crack segmentation on laser-scanned range images in this study;

iv) Through a comparison between Net-4 and CrackNet II, it is demonstrated that Net-4 outperforms the latter on both the segmentation accuracy and efficiency on the test dataset.

Furthermore, Case II is performed to demonstrate the segmentation performance of Net-4. Crack maps are generated for three roadway surfaces which contain different levels of surface variations and grooved patterns. It is shown that Net-4 achieves very high and consistent metrics values in all three roadway images. The robustness of the proposed Net-4 determined in this study is demonstrated in that it directly processes raw range images without any pre-processing procedure, and that it yields accurate and consistent segmentation performance in both Case I and II under image disturbances including grooves and surface variations.

By leveraging this DCNN-based crack segmentation methodology, Challenge 6 as described in section 1.2 can be properly addressed. However, the performance of this methodology may deteriorate under the scenario of shallow cracks, as demonstrated in section 4.3.4.

5.4 Deep Learning-Based Data Fusion for Crack Classification

The DCNN-based roadway crack classification methodology proposed in section 3.4.3 investigates the DCNN performance from the perspective of heterogeneous image data. It is focused on three objectives: *i*) demonstrating the impacts of image pre-processing, and offering suggestions regarding the use of image pre-processing in practical situations; *ii*) proposing a heterogeneous image fusion approach based on hyperspectral imaging to integrate the information in the raw intensity and range image data; and, investigating the effects of heterogeneous image data by considering four types of image data: the raw intensity, raw range, filtered range, and fused raw image data; and *iii*) proposing DCNN architectures which represent different means to exploit the fused image data, and determining the layout to yield better classification performance with data fusion.

In section 4.4, an experimental study is carried out to train and test the proposed roadway crack classification DCNNs. The experimental study consists of two cases:

Case I investigates the effect of heterogeneous image data on DCNN performance. A deep architecture which takes single-channel input, denoted as Net-A, is designed to process the raw intensity, raw range, and filtered range data; Net-B is modified from Net-A to process the fused raw image data. Besides, a benchmark architecture is evaluated using the same image data for comparison purposes. In total, seven subcases are performed in Case I. Performance metrics including Accuracy, Precision, Recall, and F1 are calculated to measure the classification performance.

i) It is clearly observed that architectures trained and tested on raw intensity image result in the worst classification performance due to low contrast issue; and, using raw range image instead of intensity image yields a significant amount of improvement on the overall classification performance;

ii) Through the comparison between the subcases using raw range image vs. filtered range image, it is shown that image pre-processing on the range image improves the performance by a noticeable margin; but considering the trade-off from the additional pre-processing procedure, it is still preferable to directly use the raw range image for analysis, especially under real-world scenarios;

iii) The use of fused raw image data yields the highest classification performance, because this approach takes advantage of the spatial co-registration feature generated from the fusion process.

It can be concluded from Case I that heterogeneous image fusion is a better alternative to image pre-processing, as an effective strategy to address the issues in range images or intensity images by cross-domain feature extraction. Through Case I, the first two objectives of this study are fulfilled;

Case II further discusses the better configuration to exploit the fused raw image data. Another deep architecture that also takes dual-channel image input is proposed in this study, denoted as Net-C. The major difference between Net-B and Net-C is that the former employs a "fuse-extract" pattern to directly extract the information from the fused raw image input, while the latter represents a "extract-fuse" pattern to first extract information from separate channels and then combine the high-level information. Performance metrics demonstrate that the "fuse-extract" scheme is more effective and efficient. Through Case II, the third objective of this study is fulfilled.

Despite the high classification performance of the proposed methodology based on heterogeneous image fusion, the resolution of detected cracks is limited, because this methodology is a patch-based approach.

5.5 Deep Learning-Based Data Fusion for Crack Segmentation

The methodology developed in section 3.4.4 explores the feasibility and strategy of heterogeneous image data fusion for DCNN-based segmentation. A novel heterogeneous image fusion strategy is proposed to address the image-related issues and uncertainties by combining the information in raw range and raw intensity image data with spatial correlation. In total, four types of image data including the raw range, raw intensity, filtered range, and fused raw image data are considered in this study. And, three encoder-decoder networks, namely, Net-1, 2, and 3, are developed to analyze these different types of image data.

Case I of the experimental study focuses on demonstrating the impacts from heterogeneous image data. Several observations and conclusions are briefly summarized as follows:

i) Among the four types of image data used for DCNN training and testing, the fused raw image data leads to the most accurate and robust segmentation performance. The advantage of exploiting the spatial co-registration feature in the fused raw image data to reduce uncertainty is demonstrated;

ii) Between the two DCNNs (Net-2 vs. Net-3) proposed to take fused raw image input, Net-3, which represents the "extract-fuse" pattern through a two-stream encoder layout, has slightly better performance than Net-2 representing the "fuse-extract" pattern;

iii) For the proposed DCNNs, image pre-processing on the raw range image is not necessary, because the performance metrics are not significantly improved upon applying image pre-processing. Also, the robustness of the proposed DCNNs against image-related issues existing in the raw range image data such as surface variations and grooved patterns is demonstrated;

iv) Through a series of comparisons, it is shown that the proposed DCNNs have the same level or higher segmentation performance than the two benchmarks, namely the VGG16-FCN and CrackNet II.

In Case II, the segmentation performance by the trained DCNNs is further demonstrated on a concrete roadway surface, which suffers from multiple image-related issues including surface variations and grooved patterns in the range image, and low intensity contrast in the intensity image. While other DCNNs expose issues such as false detections due to grooved patterns in the range image or low contrast in the intensity image, the proposed DCNNs with fused raw image input are robust against such image disturbances through cross-domain feature correlation, leading to the highest segmentation accuracy.

Through this study, it is suggested that heterogeneous data fusion be used as an effective strategy for DCNN-based roadway crack segmentation, which introduces robustness against image disturbances.

5.6 Concluding Remarks

This technical report is focused on developing an image-based condition assessment framework for civil infrastructures by leveraging advanced sensing and imaging technologies. Four research topics, as introduced in section 1.3, form a comprehensive and self-contained investigation and exploration on image-based crack detection. The proposed methodologies evolve from non-learning-based to learning-based, resulting in wider applicability and less subjectivity in cracking assessment. Meanwhile, from the perspective of image data, thorough investigations are performed through a series of comparative studies to determine the optimal strategy to exploit the image data as the fused raw image to reduce uncertainty, offering knowledge and insights to future similar applications. Regarding the detection resolution, research efforts are made to improve the resolution of the proposed methodologies from patch-level to pixel-level by developing more advanced DCNN architectures, thus leading to more accurate detection performance. Overall, all the challenges as described in section 1.2 are systematically investigated and properly addressed through this technical report.

Future research work includes: *i*) developing efficient methodologies to extract crack features such as area and width; *ii*) developing crack detection frameworks for asphalt pavements by leveraging the proposed DCNN-based methodologies; *iii*) developing robust DCNN-based crack segmentation methodologies that can distinguish cracks from other non-crack patterns including joints, shoulder drop-offs, and pavement edges.

REFERENCES

1. Total Construction Spending, U.S. Census Bureau, May 1, 2020, https://www.census.gov/construction/c30/pdf/totsa.pdf.
2. Total Construction Spending, U.S. Census Bureau, May 3, 2010, https://www.census.gov/construction/c30/pdf/pr201003.pdf.
3. Moving Ahead for Progress in the 21st Century Act (MAP-21): A Summary of Highway Provisions, Office of Policy and Governmental Affairs, Federal Highway Administration, U.S. Department of Transportation, 2012, https://www.fhwa.dot.gov/map21/summaryinfo.cfm.
4. Transportation Asset Management Plans (TAMP), Office of Asset Management, Federal Highway Administration, U.S. Department of Transportation, 2012, https://www.fhwa.dot.gov/asset/plans.cfm.
5. Highway Performance Monitoring System (HPMS), Office of Highway Policy Information, Federal Highway Administration, U.S. Department of Transportation, 1978, https://www.fhwa.dot.gov/policyinformation/hpms.cfm.
6. C. Koch, K. Georgieva, V. Kasireddy, B. Akinci, P. Fieguth, A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure, Advanced Engineering Informatics 29 (2) (2015), pp. 196-210, https://doi.org/10.1016/j.aei.2015.01.008.
7. C. Jiang, A crack detection and diagnosis methodology for automated pavement condition evaluation, Ph.D. Thesis, School of Civil and Environmental Engineering, Georgia Institute of Technology, 2015, http://hdl.handle.net/1853/55524.
8. J. G. Jang, H. Kim, T. Kim, B. Min, H. Lee, Improved flexural fatigue resistance of PVA fiber-reinforced concrete subjected to freezing and thawing cycles, Construction and Building Materials 59 (2014), pp. 129-135, https://doi.org/10.1016/j.conbuildmat.2014.02.040.
9. Y. Cheng, Y. Zhang, Y. Jiao, J. Yang, Quantitative analysis of concrete property under effects of crack, freeze-thaw and carbonation, Construction and Building Materials 129 (2016), pp. 106-115, https://doi.org/10.1016/j.conbuildmat.2016.10.113.
10. M. D. Thomas, B. Fournier, K. J. Folliard, Y. Resendez, Alkali-Silica Reactivity Field Identification Handbook, (2011), Accessed, https://rosap.ntl.bts.gov/view/dot/42676.
11. R. Adhikari, O. Moselhi, A. Bagchi, Image-based retrieval of concrete crack properties for bridge inspection, Automation in construction 39 (2014), pp. 180-194, https://doi.org/10.1016/j.autcon.2013.06.011.
12. Y. Xu, Y. Bao, J. Chen, W. Zuo, H. Li, Surface fatigue crack identification in steel box girder of bridges by a deep fusion convolutional neural network based on consumer-grade camera images, Structural Health Monitoring (2018), pp. 1475921718764873, https://doi.org/10.1177/1475921718764873.
13. E. Zalama, J. Gómez‐García‐Bermejo, R. Medina, J. Llamas, Road crack detection using visual features extracted by Gabor filters, Computer‐Aided Civil and Infrastructure Engineering 29 (5) (2014), pp. 342-358, https://doi.org/10.1111/mice.12042.
14. S. Bang, S. Park, H. Kim, H. Kim, Encoder‐decoder network for pixel‐level road crack detection in black‐box images, Computer‐Aided Civil and Infrastructure Engineering 34 (8) (2019), pp. 713-727, https://doi.org/10.1111/mice.12440.

15. J. C. Cheng, M. Wang, Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques, Automation in Construction 95 (2018), pp. 155-171, https://doi.org/10.1016/j.autcon.2018.08.006.
16. H. W. Huang, Q. T. Li, D. M. Zhang, Deep learning based image recognition for crack and leakage defects of metro shield tunnel, Tunnelling and Underground Space Technology 77 (2018), pp. 166-176, https://doi.org/10.1016/j.tust.2018.04.002.
17. M. Moore, B. M. Phares, B. Graybeal, D. Rolander, G. Washer, J. Wiss, Reliability of visual inspection for highway bridges, volume I, (2001), Accessed, https://rosap.ntl.bts.gov/view/dot/33883.
18. M. R. Jahanshahi, S. F. Masri, C. W. Padgett, G. S. Sukhatme, An innovative methodology for detection and quantification of cracks through incorporation of depth perception, Machine Vision and Applications (2013), pp. 1-15, https://doi.org/10.1007/s00138-011-0394-0.
19. Y. C. J. Tsai, F. Li, Critical assessment of detecting asphalt pavement cracks under different lighting and low intensity contrast conditions using emerging 3D laser technology, Journal of Transportation Engineering 138 (5) (2012), pp. 649-656, https://doi.org/10.1061/(ASCE)TE.1943-5436.0000353.
20. B. Peng, K. C. Wang, C. Chen, Automatic Crack Detection by Multi-Seeding Fusion on 1mm Resolution 3D Pavement Images, in: T&DI Congress 2014: Planes, Trains, and Automobiles, 2014, pp. 543-552, https://doi.org/10.1061/9780784413586.052.
21. M. R. Jahanshahi, F. Jazizadeh, S. F. Masri, B. Becerik-Gerber, Unsupervised approach for autonomous pavement-defect detection and quantification using an inexpensive depth sensor, Journal of Computing in Civil Engineering 27 (6) (2012), pp. 743-754, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000245.
22. E. Salari, G. Bao, Automated pavement distress inspection based on 2D and 3D information, in: Electro/Information Technology (EIT), 2011 IEEE International Conference on, IEEE, 2011, pp. 1-4, https://doi.org/10.1109/EIT.2011.5978575.
23. L. Bursanescu, M. Bursanescu, M. Hamdi, A. Lardigue, D. Paiement, Three-dimensional infrared laser vision system for road surface features analysis, in: ROMOPTO 2000: Sixth Conference on Optics, Vol. 4430, International Society for Optics and Photonics, 2001, pp. 801-809, https://doi.org/10.1117/12.432808.
24. H. Cheng, J.-R. Chen, C. Glazier, Y. Hu, Novel approach to pavement cracking detection based on fuzzy set theory, Journal of Computing in Civil Engineering 13 (4) (1999), pp. 270-280, https://doi.org/10.1061/(ASCE)0887-3801(1999)13:4(270).
25. K. C. P. Wang, Elements of automated survey of pavements and a 3D methodology, Journal of Modern Transportation 19 (1) (2011), pp. 51-57, https://doi.org/10.1007/BF03325740.
26. A. Zhang, Q. Li, K. C. P. Wang, S. Qiu, Matched filtering algorithm for pavement cracking detection, Transportation Research Record: Journal of the Transportation Research Board (2367) (2013), pp. 30-42, https://doi.org/10.3141/2367-04.
27. Q. Zou, Y. Cao, Q. Li, Q. Mao, S. Wang, CrackTree: Automatic crack detection from pavement images, Pattern Recognition Letters 33 (3) (2012), pp. 227-238, https://doi.org/10.1016/j.patrec.2011.11.004.
28. Y. Zhou, F. Wang, N. Meghanathan, Y. Huang, Seed-based approach for automated crack detection from pavement images, Transportation Research Record: Journal of the Transportation Research Board (2589) (2016), pp. 162-171, https://doi.org/10.3141/2589-18.

29. S. Zhou, W. Song, Robust image-based surface crack detection using range data, Journal of Computing in Civil Engineering 34 (2) (2020), pp. 04019054, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000873.
30. Z. Tong, J. Gao, H. Zhang, Innovative method for recognizing subgrade defects based on a convolutional neural network, Construction and Building Materials 169 (2018), pp. 69-82, https://doi.org/10.1016/j.conbuildmat.2018.02.081.
31. S. Zhou, W. Song, Deep learning-based roadway crack classification using laser-scanned range images: A comparative study on hyperparameter selection, Automation in Construction 114 (2020), pp. 103171, https://doi.org/10.1016/j.autcon.2020.103171.
32. M. Salman, S. Mathavan, K. Kamal, M. Rahman, Pavement crack detection using the Gabor filter, 16th International IEEE Conference on Intelligent Transportation Systems (ITSC), IEEE, 2013, pp. 2039-2044, https://doi.org/10.1109/ITSC.2013.6728529.
33. T. Nishikawa, J. Yoshida, T. Sugiyama, Y. Fujino, Concrete crack detection by multiple sequential image filtering, Computer‐Aided Civil and Infrastructure Engineering 27 (1) (2012), pp. 29-47, https://doi.org/10.1111/j.1467-8667.2011.00716.x.
34. T. Merazi-Meksen, M. Boudraa, B. Boudraa, Mathematical morphology for TOFD image analysis and automatic crack detection, Ultrasonics 54 (6) (2014), pp. 1642-1648, https://doi.org/10.1016/j.ultras.2014.03.005.
35. M. R. Jahanshahi, S. F. Masri, Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures, Automation in Construction 22 (2012), pp. 567-576, https://doi.org/10.1016/j.autcon.2011.11.018.
36. W. Ouyang, B. Xu, Pavement cracking measurements using 3D laser-scan images, Measurement Science and Technology 24 (10) (2013), pp. 105204, https://doi.org/10.1088/0957-0233/24/10/105204.
37. T. Yamaguchi, S. Hashimoto, Fast crack detection method for large-size concrete surface images using percolation-based image processing, Machine Vision and Applications 21 (5) (2010), pp. 797-809, https://doi.org/10.1007/s00138-009-0189-8.
38. I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, Cambridge, Massachusetts, 2016, ISBN: 0262337371.
39. Y. J. Cha, W. Choi, O. Büyüköztürk, Deep learning-based crack damage detection using convolutional neural networks, Computer-Aided Civil and Infrastructure Engineering 32 (5) (2017), pp. 361-378, https://doi.org/10.1111/mice.12263.
40. Y. Fei, K. C. P. Wang, A. Zhang, C. Chen, J. Q. Li, Y. Liu, G. Yang, B. Li, Pixel-level cracking detection on 3D asphalt pavement images through deep-learning-based CrackNet-V, IEEE Transactions on Intelligent Transportation Systems (2019), https://doi.org/10.1109/TITS.2019.2891167.
41. K. Zhang, H. Cheng, B. Zhang, Unified approach to pavement crack and sealed crack detection using preclassification based on transfer learning, Journal of Computing in Civil Engineering 32 (2) (2018), pp. 04018001, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000736.
42. A. Zhang, K. C. P. Wang, B. Li, E. Yang, X. Dai, Y. Peng, Y. Fei, Y. Liu, J. Q. Li, C. Chen, Automated pixel‐level pavement crack detection on 3D asphalt surfaces using a deep‐learning network, Computer‐Aided Civil and Infrastructure Engineering 32 (10) (2017), pp. 805-819, https://doi.org/10.1111/mice.12297.

43. B. Khaleghi, A. Khamis, F. O. Karray, S. N. Razavi, Multisensor data fusion: A review of the state-of-the-art, Information Fusion 14 (1) (2013), pp. 28-44, https://doi.org/10.1016/j.inffus.2011.08.001.
44. L. Wang, Heterogeneous data and big data analytics, Automatic Control and Information Sciences 3 (1) (2017), pp. 8-15, http://pubs.sciepub.com/acis/3/1/3.
45. W. Elmenreich, R. Leidenfrost, Fusion of heterogeneous sensors data, in: 2008 International Workshop on Intelligent Solutions in Embedded Systems, IEEE, 2008, pp. 1-10, https://doi.org/10.1109/CISP.2010.5647496.
46. I. Abdel-Qader, O. Abudayyeh, M. E. Kelly, Analysis of edge-detection techniques for crack identification in bridges, Journal of Computing in Civil Engineering 17 (4) (2003), pp. 255-263, https://doi.org/10.1061/(ASCE)0887-3801(2003)17:4(255).
47. S. Hashimoto, Practical image measurement of crack width for real concrete structure, Electronics and Communications in Japan 92 (10) (2009), pp. 1-12, https://doi.org/10.1002/ecj.10151.
48. Z. Zhu, S. German, I. Brilakis, Visual retrieval of concrete crack properties for automated post-earthquake structural safety evaluation, Automation in Construction 20 (7) (2011), pp. 874-883, https://doi.org/10.1016/j.autcon.2011.03.004.
49. Y. C. J. Tsai, V. Kaul, R. M. Mersereau, Critical assessment of pavement distress segmentation methods, Journal of transportation engineering 136 (1) (2009), pp. 11-19, https://doi.org/10.1061/(ASCE)TE.1943-5436.0000051.
50. O. Alekseychuk, Detection of crack-like indications in digital radiography by global optimisation of a probabilistic estimation function, (2006), https://inis.iaea.org/collection/NCLCollectionStore/_Public/37/079/37079015.pdf.
51. Y. Huang, B. Xu, Automatic inspection of pavement cracking distress, Journal of Electronic Imaging 15 (1) (2006), pp. 013017, https://doi.org/10.1117/1.2177650.
52. Y. Fujita, Y. Hamamoto, A robust automatic crack detection method from noisy concrete surfaces, Machine Vision and Applications 22 (2) (2011), pp. 245-254, https://doi.org/10.1007/s00138-009-0244-5.
53. Y. Huang, Y. C. J. Tsai, Dynamic programming and connected component analysis for an enhanced pavement distress segmentation algorithm, Transportation Research Record: Journal of the Transportation Research Board (2225) (2011), pp. 89-98, https://doi.org/10.3141/2225-10.
54. L. Li, L. Sun, G. Ning, S. Tan, Automatic pavement crack recognition based on BP neural network, PROMET-Traffic&Transportation 26 (1) (2014), pp. 11-22, https://doi.org/10.7307/ptt.v26i1.1477.
55. K. C. P. Wang, W. Gong, R. P. Elliott, A feasibility study on data automation for comprehensive pavement condition survey, in: Proc. of 6th Internat. Conference on Managing Pavements., 2004, https://www.researchgate.net/profile/Kelvin_Wang3/publication/267238769_A_Feasibility_Study_on_Data_Automation_for_Comprehensive_Pavement_Condition_Survey/links/555c772608aec5ac22331d36.pdf.
56. Z. Hou, K. C. P. Wang, W. Gong, Experimentation of 3D pavement imaging through stereovision, in: International Conference on Transportation Engineering 2007, 2007, pp. 376-381, https://doi.org/10.1061/40932(246)62.

57. K. C. P. Wang, W. Gong, Automated pavement distress survey: a review and a new direction, in: Pavement Evaluation Conference, 2002, pp. 21-25, https://www.researchgate.net/profile/Kelvin_Wang3/publication/238694797_Automated_Pavement_Distress_Survey_A_Review_and_A_New_Direction/links/555c772808ae91e75e774088.pdf.

58. M. Torok, M. Fard, K. Kochersberger, Post-Disaster robotic building assessment: Automated 3D crack detection from image-based reconstructions, in: International Conference on Computing in Civil Engineering, 2012, pp. 397-404, https://doi.org/10.1061/9780784412343.0050.

59. M. R. Jahanshahi, S. F. Masri, A new methodology for non-contact accurate crack width measurement through photogrammetry for automated structural safety evaluation, Smart Materials and Structures 22 (3) (2013), pp. 035019, https://doi.org/10.1088/0964-1726/22/3/035019.

60. K. N. Snavely, Scene Reconstruction and Visualization from Internet Photo Collections, University of Washington, 2008, https://www.cs.cornell.edu/~snavely/publications/thesis/thesis.pdf.

61. Y. C. J. Tsai, C. Jiang, Z. Wang, Pavement crack detection using high-resolution 3D line laser imaging technology, in: 7th RILEM International Conference on Cracking in Pavements: Mechanisms, Modeling, Testing, Detection and Prevention Case Histories, Springer Netherlands, Dordrecht, 2012, pp. 169-178, https://doi.org/10.1007/978-94-007-4566-7_17.

62. Y. C. J. Tsai, A. Chatterjee, C. Jiang, Challenges and lessons from the successful implementation of automated road condition surveys on a large highway system, Signal Processing Conference (EUSIPCO), 2017 25th European, IEEE, 2017, pp. 2031-2035, https://doi.org/10.23919/EUSIPCO.2017.8081566.

63. J. Laurent, D. Lefebvre, E. Samson, Development of a new 3D transverse laser profiling system for the automatic measurement of road cracks, in: Symposium on Pavement Surface Characteristics, 6th, 2008, Portoroz, Slovenia., 2008, https://trid.trb.org/view/1151548.

64. A. Zhang, K. C. P. Wang, R. Ji, Q. J. Li, Efficient system of cracking-detection algorithms with 1-mm 3D-surface models and performance measures, Journal of Computing in Civil Engineering 30 (6) (2016), pp. 04016020, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000581.

65. C. Jiang, Y. C. J. Tsai, Enhanced crack segmentation algorithm using 3D pavement data, Journal of Computing in Civil Engineering 30 (3) (2015), pp. 04015050, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000526.

66. B. J. Lee, H. D. Lee, Position ‐ invariant neural network for digital pavement crack analysis, Computer ‐ Aided Civil and Infrastructure Engineering 19 (2) (2004), pp. 105-118, https://doi.org/10.1111/j.1467-8667.2004.00341.x.

67. T. Saar, O. Talvik, Automatic asphalt pavement crack detection and classification using neural networks, in: 12th Biennial Baltic Electronics Conference, Tallinn, Estonia, 2010, pp. 345-348, https://doi.org/10.1109/BEC.2010.5630750.

68. G. Moussa, K. Hussain, A new technique for automatic detection and parameters estimation of pavement crack, in: 4th International Multi-Conference on Engineering Technology Innovation, Orlando, Florida, 2011, https://pdfs.semanticscholar.org/0cdc/86650ec3dbed96e01a2bc5ecac05e6ab6002.pdf.

69. D. Xie, L. Zhang, L. Bai, Deep learning in visual computing and signal processing, Applied Computational Intelligence and Soft Computing 2017 (2017), https://doi.org/10.1155/2017/1320780.
70. L. Zhang, F. Yang, Y. D. Zhang, Y. J. Zhu, Road crack detection using deep convolutional neural network, in: 2016 IEEE International Conference on Image Processing, Phoenix, Arizona, 2016, pp. 3708-3712, https://doi.org/10.1109/ICIP.2016.7533052.
71. S. Park, S. Bang, H. Kim, H. Kim, Patch-based crack detection in black box images using convolutional neural networks, Journal of Computing in Civil Engineering 33 (3) (2019), pp. 04019017, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000831.
72. R. Girshick, Fast r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440-1448, http://openaccess.thecvf.com/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf.
73. S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: Advances in neural information processing systems, 2015, pp. 91-99, http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf.
74. Y. J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, O. Büyüköztürk, Autonomous structural visual inspection using region ‐ based deep learning for detecting multiple damage types, Computer ‐ Aided Civil and Infrastructure Engineering 33 (9) (2018), pp. 731-747, https://doi.org/10.1111/mice.12334.
75. H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, H. Omata, Road damage detection using deep neural networks with images captured through a smartphone, arXiv preprint arXiv:1801.09454 (2018), https://arxiv.org/abs/1801.09454.
76. Y. Xue, Y. Li, A fast detection method via region ‐ based fully convolutional neural networks for shield tunnel lining defects, Computer ‐ Aided Civil and Infrastructure Engineering 33 (8) (2018), pp. 638-654, https://doi.org/10.1111/mice.12367.
77. X. Yang, H. Li, Y. Yu, X. Luo, T. Huang, X. Yang, Automatic pixel ‐ level crack detection and measurement using fully convolutional network, Computer ‐ Aided Civil and Infrastructure Engineering 33 (12) (2018), pp. 1090-1109, https://doi.org/10.1111/mice.12412.
78. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014), https://arxiv.org/abs/1409.1556.
79. Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, S. Wang, Deepcrack: Learning hierarchical convolutional features for crack detection, IEEE Transactions on Image Processing 28 (3) (2018), pp. 1498-1512, https://doi.org/10.1109/TIP.2018.2878966.
80. V. Badrinarayanan, A. Handa, R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling, arXiv preprint arXiv:1505.07293 (2015).
81. O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical image computing and computer-assisted intervention, Springer, 2015, pp. 234-241, https://arxiv.org/pdf/1505.04597.pdf.
82. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 30th IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, Hawaii, 2016, pp.

770-778, http://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf.

83. C. V. Dung, Autonomous concrete crack detection using deep fully convolutional neural network, Automation in Construction 99 (2019), pp. 52-58, https://doi.org/10.1016/j.autcon.2018.11.028.

84. C. Tan, N. Uddin, Y. M. Mohammed, Deep Learning-Based Crack Detection Using Mask R-CNN Technique, in: 9th International Conference on Structural Health Monitoring of Intelligent Infrastructure, 2019, https://www.researchgate.net/profile/Chengjun_Tan/publication/337885795_Deep_Learning-Based_Crack_Detection_Using_Mask_R-CNN_Technique/links/5df0771ba6fdcc2837177db4/Deep-Learning-Based-Crack-Detection-Using-Mask-R-CNN-Technique.pdf.

85. K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961-2969, http://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf.

86. Y. Bengio, I. Goodfellow, A. Courville, Deep learning, Citeseer, 2017.

87. H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, R. M. Summers, Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning, IEEE transactions on medical imaging 35 (5) (2016), pp. 1285-1298, https://doi.org/10.1109/TMI.2016.2528162.

88. V. Dumoulin, F. Visin, A guide to convolution arithmetic for deep learning, arXiv preprint arXiv:1603.07285 (2016).

89. S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167 (2015).

90. A. L. Maas, A. Y. Hannun, A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: 30th International Conference on Machine Learning, Vol. 30, No. 1, Atlanta, Georgia, 2013, pp. 3, http://robotics.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf.

91. V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 807-814, https://www.cs.toronto.edu/~hinton/absps/reluICML.pdf.

92. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (1) (2014), pp. 1929-1958, https://dl.acm.org/doi/abs/10.5555/2627435.2670313.

93. M. Mohtasham Khani, S. Vahidnia, L. Ghasemzadeh, Y. E. Ozturk, M. Yuvalaklioglu, S. Akin, N. K. Ure, Deep-learning-based crack detection with applications for the structural health monitoring of gas turbines, Structural Health Monitoring (2019), pp. 1475921719883202, https://doi.org/10.1177/1475921719883202.

94. B. Kim, S. Cho, Automated vision-based detection of cracks on concrete surfaces using a deep learning technique, Sensors 18 (10) (2018), pp. 3452, https://doi.org/10.3390/s18103452.

95. A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, Lake Tahoe,

Nevada, 2012, pp. 1097-1105, http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

96. K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the devil in the details: Delving deep into convolutional nets, arXiv preprint arXiv:1405.3531 (2014).
97. A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, J. Garcia-Rodriguez, A review on deep learning techniques applied to semantic segmentation, arXiv preprint arXiv:1704.06857 (2017), https://arxiv.org/abs/1704.06857.
98. C. M. Bishop, Pattern recognition and machine learning, springer, 2006, 1493938436.
99. Y. Bengio, Practical recommendations for gradient-based training of deep architectures, Neural Networks: Tricks of the Trade, Springer, 2012, pp. 437-478.
100. X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: 13th International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 2010, pp. 249-256, http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf?source=post_page.
101. A. E. Orhan, X. Pitkow, Skip connections eliminate singularities, arXiv preprint arXiv:1701.09175 (2017).
102. G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700-4708, https://doi.org/10.1109/CVPR.2017.243.
103. N. Japkowicz, S. Stephen, The class imbalance problem: A systematic study, Intelligent data analysis 6 (5) (2002), pp. 429-449, https://doi.org/10.3233/IDA-2002-6504.
104. T. Fawcett, An introduction to ROC analysis, Pattern Recognition Letters 27 (8) (2006), pp. 861-874.
105. P. N. Tan, Introduction to data mining, Pearson Education India, 2018, ISBN: 813176463X.
106. G. Csurka, D. Larlus, F. Perronnin, F. Meylan, What is a good evaluation measure for semantic segmentation?, in: BMVC, Vol. 27, Citeseer, 2013, pp. 2013, http://dx.doi.org/10.5244/C.27.32.
107. G. Shrivakshan, C. Chandrasekar, A comparison of various edge detection techniques used in image processing, IJCSI International Journal of Computer Science Issues 9 (5) (2012), pp. 272-276, https://www.ijcsi.org/papers/IJCSI-9-5-1-269-276.pdf.
108. M. Sabin, Contouring—the state of the art, Fundamental algorithms for computer graphics, Springer, 1985, pp. 411-482, https://link.springer.com/chapter/10.1007/978-3-642-84574-1_20.
109. D. Watson, Contouring: a guide to the analysis and display of spatial data, Elsevier, 2013, 1483287351.
110. R. Shumway, Statistics and data analysis in geology, Taylor & Francis, 1987.
111. F. P. Agterberg, Trend surface analysis, Spatial statistics and models, Springer, 1984, pp. 147-171, https://link.springer.com/chapter/10.1007/978-94-017-3048-8_8.
112. H. Elbehiery, A. Hefnawy, M. Elewa, Surface Defects Detection for Ceramic Tiles Using Image Processing and Morphological Techniques, in: WEC (5), 2005, pp. 158-162, https://www.researchgate.net/publication/221017750_Surface_Defects_Detection_for_Ceramic_Tiles_Using_Image_Processing_and_Morphological_Techniques.
113. X. J. Xu, X. N. Zhang, Crack detection of reinforced concrete bridge using video image, Journal of Central South University 20 (9) (2013), pp. 2605-2613, https://doi.org/10.1007/s11771-013-1775-5.

114. M. Rabah, A. Elhattab, A. Fayad, Automatic concrete cracks detection and mapping of terrestrial laser scan data, NRIAG Journal of Astronomy and Geophysics 2 (2) (2013), pp. 250-255, https://doi.org/10.1016/j.nrjag.2013.12.002.

115. K. Wang, L. Lin, Q. J. Li, V. Nguyen, G. Hayhoe, A. Larkin, Runway groove identification and evaluation using 1 mm 3D image data, Airfield and Highway Pavement 2013: Sustainable and Efficient Pavements, 2013, pp. 730-741, https://doi.org/10.1061/9780784413005.059.

116. D. Sundararajan, Digital Image Processing: A Signal Processing and Algorithmic Approach, Springer, 2017, 9811061130.

117. E. R. Davies, Computer and machine vision: theory, algorithms, practicalities, Academic Press, 2012, 0123869919.

118. M. A. Farooque, J. S. Rohankar, Survey on various noises and techniques for denoising the color image, International Journal of Application or Innovation in Engineering & Management (IJAIEM) 2 (11) (2013), pp. 217-221, https://www.ijaiem.org/volume2issue11/IJAIEM-2013-11-24-070.pdf.

119. A. Mohan, S. Poobal, Crack detection using image processing: A critical review and analysis, Alexandria Engineering Journal (2017), https://doi.org/10.1016/j.aej.2017.01.020.

120. L. G. Mosher, Results from studies of highway grooving and texturing by serveral state highway departments, Pavement grooving and traction studies 5073 (1969), pp. 465, https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19690011116.pdf.

121. E. E. Farnsworth, Pavement grooving on highways, Pavement Grooving and Traction Studies 5073 (1969), pp. 411, https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19690011114.pdf.

122. G. P. Ong, T. Fwa, Transverse pavement grooving against hydroplaning. I: Simulation model, Journal of transportation engineering 132 (6) (2006), pp. 441-448, https://doi.org/10.1061/(ASCE)0733-947X(2006)132:6(441).

123. M. Nieniewski, L. Chmielewski, A. Jozwik, M. Sklodowski, Morphological detection and feature-based classification of cracked cegions in ferrites, Machine Graphics and Vision 8 (4) (1999), pp. 699-712, https://pdfs.semanticscholar.org/8cd2/a6efce0221c095e434a710485997b81723f2.pdf.

124. X. Yu, X. Chen, W. Chen, Image smoothing algorithm for grooved cement concrete pavement based on unidirectional total variation and Curvelet transform, in: Computer and Communications (ICCC), 2016 2nd IEEE International Conference on, IEEE, 2016, pp. 713-717, https://doi.org/10.1109/CompComm.2016.7924795.

125. M. P. Cipolletti, C. A. Delrieux, G. M. Perillo, M. C. Piccolo, Superresolution border segmentation and measurement in remote sensing images, Computers & geosciences 40 (2012) (2012), pp. 87-96, https://doi.org/10.1016/j.cageo.2011.07.015.

126. C. Maple, Geometric design and space planning using the marching squares and marching cube algorithms, Geometric Modeling and Graphics, 2003. Proceedings. 2003 International Conference on, IEEE, 2003, pp. 90-95, https://doi.org/10.1109/GMAG.2003.1219671.

127. W. K. Pratt, Digital image processing, John Wiley & Sons, Inc., 2001.

128. S. Faghih-Roohi, S. Hajizadeh, A. Núñez, R. Babuska, B. De Schutter, Deep convolutional neural networks for detection of rail surface defects, in: 2016 IEEE International Joint

Conference on Neural Networks, Vancouver, British Columbia, Canada, 2016, pp. 2584-2589, https://doi.org/10.1109/IJCNN.2016.7727522.

129. L. Pauly, D. Hogg, R. Fuentes, H. Peel, Deeper networks for pavement crack detection, in: 34th International Symposium on Automation and Robotics in Construction, Taipei, Taiwan, 2017, pp. 479-485, http://eprints.whiterose.ac.uk/120380/1/ISARC2017-Paper066.pdf.
130. W. Song, S. Zhou, Laser-scanned roadway range image dataset (LRRD), (2020), Accessed, https://doi.org/10.17603/ds2-bzv3-nc78.
131. P. Wang, H. Huang, Comparison analysis on present image-based crack detection methods in concrete structures, in: 2010 3rd International Congress on Image and Signal Processing (CISP), Vol. 5, IEEE, 2010, pp. 2530-2533, https://doi.org/10.1109/CISP.2010.5647496.
132. R. T. Wu, M. R. Jahanshahi, Data fusion approaches for structural health monitoring and system identification: past, present, and future, Structural Health Monitoring (2018), pp. 1475921718798769, https://doi.org/10.1177/1475921718798769.
133. Y. Feng, Z. Zhang, X. Zhao, R. Ji, Y. Gao, Gvcnn: Group-view convolutional neural networks for 3d shape recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 264-272.
134. F. C. Chen, M. R. Jahanshahi, R. T. Wu, C. Joffe, A texture - based video processing methodology using Bayesian data fusion for autonomous crack detection on metallic surfaces, Computer - Aided Civil and Infrastructure Engineering 32 (4) (2017), pp. 271-287, https://doi.org/10.1111/mice.12256.
135. F. C. Chen, M. R. Jahanshahi, NB-CNN: Deep learning-based crack detection using convolutional neural network and Naïve Bayes data fusion, IEEE Transactions on Industrial Electronics 65 (5) (2018), pp. 4392-4400, https://doi.org/10.1109/TIE.2017.2764844.
136. R. Heideklang, P. Shokouhi, Multi-sensor image fusion at signal level for improved near-surface crack detection, NDT & E International 71 (2015), pp. 16-22, https://doi.org/10.1016/j.ndteint.2014.12.008.
137. G. H. Beckman, D. Polyzois, Y. J. Cha, Deep learning-based automatic volumetric damage quantification using depth camera, Automation in Construction 99 (2019), pp. 114-124, https://doi.org/10.1016/j.autcon.2018.12.006.
138. D. Landgrebe, Hyperspectral image data analysis, IEEE Signal Processing Magazine 19 (1) (2002), pp. 17-28, https://doi.org/10.1109/79.974718.
139. S. Sorncharean, S. Phiphobmongkol, Crack detection on asphalt surface image using enhanced grid cell analysis, in: Electronic Design, Test and Applications, 2008. DELTA 2008. 4th IEEE International Symposium on, IEEE, 2008, pp. 49-54, https://doi.org/10.1109/DELTA.2008.101.
140. J. Huang, W. Liu, X. Sun, A Pavement Crack Detection Method Combining 2D with 3D Information Based on Dempster - Shafer Theory, Computer - Aided Civil and Infrastructure Engineering 29 (4) (2014), pp. 299-313, https://doi.org/10.1111/mice.12041.
141. D. Zhang, Q. Li, Y. Chen, M. Cao, L. He, B. Zhang, An efficient and reliable coarse-to-fine approach for asphalt pavement crack detection, Image and Vision Computing 57 (2017), pp. 130-146, https://doi.org/10.1016/j.imavis.2016.11.018.

142. M. A. Shahin, H. R. Maier, M. B. Jaksa, Data division for developing neural networks applied to geotechnical engineering, Journal of Computing in Civil Engineering 18 (2) (2004), pp. 105-114, https://doi.org/10.1061/(ASCE)0887-3801(2004)18:2(105).

143. MATHWORKS, MATLAB deep learning toolbox, (2019), Accessed, https://www.mathworks.com/products/deep-learning.html.

144. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: 30th IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, Hawaii, 2016, pp. 2818-2826, https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.pdf.

145. M. Thoma, Analysis and optimization of convolutional neural network architectures, arXiv preprint arXiv:1707.09725 (2017).

146. Y. z. Lin, Z. h. Nie, H. w. Ma, Structural damage detection with automatic feature - extraction through deep learning, Computer - Aided Civil and Infrastructure Engineering 32 (12) (2017), pp. 1025-1046, https://doi.org/10.1111/mice.12313.

147. C. Nwankpa, W. Ijomah, A. Gachagan, S. Marshall, Activation functions: Comparison of trends in practice and research for deep learning, arXiv preprint arXiv:1811.03378 (2018).

148. N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P. T. P. Tang, On large-batch training for deep learning: Generalization gap and sharp minima, arXiv preprint arXiv:1609.04836 (2016).

149. C. Xing, D. Arpit, C. Tsirigotis, Y. Bengio, A walk with sgd, arXiv preprint arXiv:1802.08770 (2018).

150. S. Jastrzębski, D. Arpit, N. Ballas, V. Verma, T. Che, Y. Bengio, Residual connections encourage iterative inference, arXiv preprint arXiv:1710.04773 (2017).

151. L. N. Smith, A disciplined approach to neural network hyper-parameters: Part 1--learning rate, batch size, momentum, and weight decay, arXiv preprint arXiv:1803.09820 (2018).

152. Z. Liu, Y. Cao, Y. Wang, W. Wang, Computer vision-based concrete crack detection using U-net fully convolutional networks, Automation in Construction 104 (2019), pp. 129-139, https://doi.org/10.1016/j.autcon.2019.04.005.

APPENDIX A: LIST OF PUBLICATIONS

This section lists the journal papers that have been published, which are based on the research outcomes from this technical report.

S. Zhou, W. Song, Crack segmentation through deep convolutional neural networks and heterogeneous image fusion, *Automation in Construction*, 125 (2020): p. 103605, https://doi.org/10.1016/j.autcon.2021.103605. (This manuscript is corresponding to sections 3.4, 4.5, and 5.5 of this technical report).

S. Zhou, W. Song, Concrete roadway crack segmentation using encoder-decoder networks with range images, *Automation in Construction*, 120 (2020): p. 103403, https://doi.org/10.1016/j.autcon.2020.103403. (This manuscript is corresponding to sections 3.3, 4.3, and 5.3 of this technical report).

S. Zhou, W. Song, Deep learning-based roadway crack classification with heterogeneous image data fusion, *Structural Health Monitoring*, (2020): p. 1475921720948434, https://doi.org/10.1177/1475921720948434. (This manuscript is corresponding to sections 3.4, 4.4, and 5.4 of this technical report).

S. Zhou, W. Song, Deep learning-based roadway crack classification using laser-scanned range images: A comparative study on hyperparameter selection, *Automation in Construction*, 114 (2020): p. 103171, https://doi.org/10.1016/j.autcon.2020.103171. (This manuscript is corresponding to sections 3.2, 4.2, and 5.2 of this technical report).

S. Zhou, W. Song, Robust image-based surface crack detection using range data, *Journal of Computing in Civil Engineering*, 34 (2) (2020): p. 04019054, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000873. (This manuscript is corresponding to sections 3.1, 4.1, and 5.1 of this technical report).