IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 August 2023, accepted 23 October 2023, date of publication 27 October 2023, date of current version 6 November 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3328243*

RESEARCH ARTICLE

# Efficient Dataset Collection for Concrete Crack Detection With Spatial-Adaptive Data Augmentation

**JONG-HYUN KIM[1] AND JUNG LEE[2]**
[1]Department of Design Technology, College of Software and Convergence, Inha University, Michuhol-gu, Incheon 22212, South Korea
[2]Department of Computer Engineering, Hanbat National University, Yuseong-gu, Daejeon 34158, South Korea

Corresponding author: Jung Lee (airjung@hanbat.ac.kr)

This work was supported in part by the Hanbat National University, in 2023 (Contribution Rate: 20%); in part by the National Research Foundation of Korea (NRF) funded by the Korean Government (MSIT) under Grant RS-2023-00254695 (Contribution Rate: 10%); and in part by the Basic Science Research Program through the NRF funded by the Ministry of Education under Grant 2022R1F1A1063180 (Contribution Rate: 70%).

**ABSTRACT** In this paper, we propose a data augmentation technique based on Convolutional Neural Networks (CNN or ConvNet) training to efficiently obtain a dataset of images containing concrete cracks. Concrete cracks usually do not have a standardized shape and have complex patterns, making it difficult to obtain images of them, and there is a risk of exposure to dangerous situations when securing data. Therefore, in this paper, we efficiently address the difficulty of dataset collection by using a data augmentation technique based on learning the direction and thickness of cracks, which is cost-effective and time-efficient. Moreover, to improve efficiency, we introduce a method of adaptively handling crack data by constructing a quadtree based on the presence of cracks. To confirm the extent of the improvement in accuracy, we conducted experiments applying the crack detection algorithm to various scenes, and the accuracy was improved in all scenes when measured by IoU (Intersection over union) accuracy. When the algorithm was performed without augmenting the crack data, the false detection rate was about 25%. However, when we augmented the data using our method, the false detection rate significantly decreased to 3%.

**INDEX TERMS** Spatial-adaptive augmentation, concrete crack, data augmentation, convolutional neural networks, crack direction, crack thickness.

## I. INTRODUCTION

In this paper, we aim to provide means from a computer science perspective to prevent building collapse incidents that often occur worldwide. When workers search for concrete cracks on site, there is a risk of exposure to dangerous situations, and it takes a lot of time and cost. Moreover, the accuracy of finding cracks is limited by conditions such as line of sight and weather during work [1]. In this paper, we propose a method for efficiently augmenting concrete crack data based on direction and thickness using neural networks, with the aim of improving the accuracy of crack detection.

The associate editor coordinating the review of this manuscript and approving it for publication was Lei Wei.

One of the most common problems that occur when using concrete, the most widely used material in building construction, is cracking. As buildings age, the number and size of cracks increase, which adversely affects the safety and durability of the structure. Therefore, it is crucial to regularly detect cracks in concrete structures and take corresponding maintenance measures for the safety and durability of the structures [2], [3], [4]. Typically, cracks in structures are detected by experts equipped with specialized equipment who physically inspect the site. However, this method requires a lot of labor and time, and in sites with large apartments, the accuracy of crack detection can decrease, eventually posing a danger to people's safety.

To efficiently and reliably detect cracks and overcome the drawbacks of manual detection, many studies have utilized

image processing techniques [5]. In the field of computer vision, various research has been conducted for automatic detection of cracks in images : image thresholding [6], [7], edge detection [8], wavelet transformation [9], [10], and machine learning [11]. The image thresholding method classifies cracks at the pixel level based on various pixel values, making it easy to simplify the image and perform post-processing. Edge detection method efficiently detects cracks in images by applying differential operators to the image [12]. The basic concept of wavelet transformation is a function, such as an image signal, and it uses wavelet functions or basis function sets to find cracks based on this signal [13]. Machine learning extracts feature vectors of cracks from training data and infers results through learning. Although these methods can efficiently detect cracks, research is actively being conducted to improve the non-uniformity of cracks, the diversity of surface textures, and the complexity of backgrounds.

Deep learning techniques can learn deep nonlinear network structures and model abstract representations. Zhang et al. proposed a crack detection technique based on deep learning [14], which is one of the early studies that applied CNN to road crack detection. The pavement road photos used in this study were captured by a smartphone, and the network model was designed based on the Caffe deep learning framework. Furthermore, the effectiveness of the deep learning technique was demonstrated by comparing it with existing machine learning classifiers such as Support Vector Machine (SVM) and Boost methods. Pauly et al. studied the effect of the depth of the CNN and the positional variation between the training and testing data on the accuracy of the pavement crack detection technique [15]. The results showed that increasing the depth of the network improves accuracy, while significant decrease in detection accuracy occurs when the position of the image is changed. Maeda et al. generated a large-scale road damage dataset and marked the location and type of road damage in each image [16]. They then trained a damage detection model using an end-to-end object detection method based on deep learning. They applied this method to a mobile app to enable easy detection of road damage even in areas with a shortage of experts and labor.

Xu et al. built an end-to-end detection model to automatically detect cracks in bridges [17]. They demonstrated that using depth-wise separable convolutions can effectively reduce the number of parameters [26]. Li et al. proposed the YOLOv3-Lite method to improve speed without compromising detection accuracy [18]. Tong et al. proposed a method for efficiently detecting damages in low-cost by automatically detecting and measuring surface-penetrating images using CNN and reconstructing hidden cracks in 3D [19]. Yang et al. proposed a technique based on FCN (Fully Convolutional Network) to detect cracks at the pixel level [20]. FCN consists of upsampling and downsampling layers that can detect objects of various scales, and the accuracy of crack detection was measured to be 97.96%. Zhu and Song improved VGG16 using transfer learning and modeled accurate classification of surface defects in concrete bridges [21]. Deng et al. added an area-based deformation module to Faster R-CNN [22], R-FCN [23], and FPN-based Faster R-CNN [24] to improve the accuracy of crack detection [25]. As mentioned earlier, most studies have modeled surface crack detection in concrete using various transfer learning techniques to improve the accuracy of crack detection. Using transfer learning makes model training easier and faster, but this is a characteristic of the network architecture. There are no advantages to preprocessing or data augmentation methods for crack data. In this paper, we propose a new network training model for crack data augmentation, taking into account the limitations of using transfer learning. We demonstrate through experiments that our proposed model can have a positive impact on crack detection.

## II. RELATED WORK

In this section, we will examine the crack detection network method, data augmentation techniques for it, and image data augmentation techniques related to my research.

### A. NETWORKS FOR CRACK DETECTION

Handling issues such as uneven lighting, stained areas, blurriness, noise, etc. is crucial in image-based crack detection research. One way to detect cracks is to define a feature that clearly distinguishes between noise and cracks, such as the iterative clipping method that assumes cracks are generally darker than their surroundings [26]. Li et al. proposed the NDHM (Neighboring Difference Histogram) method to segment crack images [27]. However, this method can malfunction when there are many dark areas, such as shadows, in the test image. A wavelet transform-based method was proposed to enhance the contrast of crack regions in order to improve the accuracy and completeness of crack detection [28], [29]. However, due to the anisotropic nature of wavelets, they often fail to handle cracks with high curvature or poor continuity. The FoSA (F* seed-growing approach) method was proposed to extend the NDHM with more powerful noise reduction capabilities [30]. The CrackTree method was designed to remove the shadow cast on cracks, but it takes a long time to compute [31]. Approaches that measure the anisotropy of images, such as CTA (Conditional Texture Anisotropy) [32] and FFA (Free-Form Anisotropy) [33], which consider both brightness and connectivity, have shown good results in crack detection. However, they are often sensitive to edge areas and can result in noise. Methods have been proposed to address these issues using saliency [34].

Hu et al. used LBP(Local binary pattern) method to obtain good crack detection results through local features [35], but parameter adjustment is required for each

image. Zhang et al. proposed a BTH (Black Top-Hat) transformation and threshold segmentation method to detect cracks on concrete tunnel surfaces [36], but this method does not work well when the lighting is uneven. Recently, Zhang et al. modeled the Region of Aggregation (ROA) for considering multiple data such as spatial distribution, intensity, and geometric features of cracks, and the Region of Belief (ROB) for expanding the crack area [7]. This method produces good results for thin-shaped cracks.

### B. IMAGE DATA AUGMENTATION

Generally, data augmentation is a technique of applying various transformations to original data to increase the amount of data used for learning-based research. Data augmentation is typically used during the training phase but it can also be used during the test phase, which is known as Test-time augmentation (TTA). This method augments a single test image into multiple images and performs inference on them. The outputs generated from this process are then ensembled. This technique is commonly used in competitions such as Kaggle.

#### 1) IMAGE MANIPULATION-BASED APPROACH

There are approaches that utilize pixel-level transform and spatial-level transform techniques. Pixel-level transform is a technique that applies blur, jitter, noise, and other effects to an image to transform it at the pixel level. Various techniques such as Gaussian blur, motion blur, brightness jitter, contrast jitter, saturation jitter, ISO noise, JPEG compression, and so on are used for pixel-level transforms. Spatial-level transform refers to techniques that transform the image space. Representative methods include flip and rotation, and cropping, which involves cutting out a portion of the image, is also frequently used.

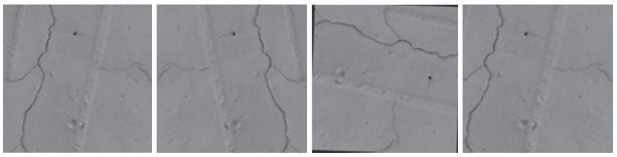
Kang et al. proposed a technique that randomly shuffles the feature values within a non-overlapping sliding window [46]. Applying this method improves performance, but the performance is significantly influenced by the size of the window. Hiroshi Inoue proposed a method that randomly selects two images from the training set, randomly crops them to a size of 224 $\times$ 224, and applies random horizontal flip to them [47]. The obtained two patches are averaged to create a mixed patch. Since two images are mixed but only one label is used, the accuracy may be compromised. Summers and Dinneen proposed an improved method for mixing two images, departing from the simple average blending approach. Instead, they introduced eight new mixing methods to enhance the process [48]. Zhang et al. proposed a technique where two images are interpolated using weighted linear interpolation with lambda values ranging from 0 to 1 [49]. This method is simple but it can improve the generalization performance of the model, prevent memorization of corrupt labels, and make the model more sensitive to adversarial examples.

Takahashi et al. proposed the RICAP (Random image cropping patching) technique, which combines randomly cropped patches from four images to create a single image [50]. Additionally, similar to the mixup technique, Takahashi et al. mixed the labels of the four images based on the area ratio of the patches, creating soft labels for training. However, in the cropping process, if the background rather than the main region of the image is cropped, it can lead to incorrect labels being assigned. Verma et al. proposed a method of handling mixup at the hidden representation or feature map level instead of the input image level [51]. By using this method, not only can the decision boundary be smoothed, but also the advantages of mixup can be preserved. Unlike the previously mentioned methods that involve mixing images, Zhong et al. chose an approach that involves erasing image regions [52]. This method involves creating a random-sized boundary box on the input image and filling it with random noise, ImageNet mean value, 0, 255, or other values before using it for training.

Singh et al. proposed a method where the image is divided into a grid, and patches are randomly erased at each iteration during training [53]. This approach allows the network to consider various parts of the image rather than focusing only on specific parts of the object, leading to more comprehensive predictions. Yun et al. proposed a method called CutMix, which combines Mixup and Cutout techniques [54]. This technique involves selecting a box area on an image and erasing it, then filling the empty region with a patch extracted from another image. The labels are mixed in proportion to the area of the patch. Hendrycks et al. proposed a method called Augmix, where multiple augmentation techniques are applied to an image either in series or in parallel, and then mixed with the original image [55]. This method was proposed not primarily to improve general test accuracy, but to enhance performance on datasets such as ImageNet-C and ImageNet-P, which are designed to measure robustness. To alleviate the issue of strong edges resulting from the process of cutting and pasting patches in CutMix, Lee et al. proposed a method called SmoothMix, which smooths the boundary regions to create a more gradual transition [56]. Kim et al. proposed a method called PuzzleMix, which blends images while preserving their saliency. This approach aims to maintain the salient regions of the original images while creating a blended output [57]. This enables the preservation of local statistics in each image and exhibits better generalization performance compared to existing mixing techniques. It also enhances robustness against adversarial attacks.

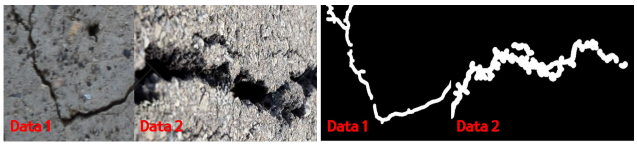
### C. DATASETS AND DATA AUGMENTATION FOR CRACK DETECTION

The dataset collected by Li and Zhao is widely used in crack detection research [37]. The dataset consists of images



(a) (b) (c) (d)

**FIGURE 1.** Images after data augmentation : (a) original image, (b) horizontal flip, (c) rotation, (d) horizontal shift.




(a) Input data (b)

**FIGURE 2.** Extracted skeleton from crack image.

captured using a smartphone on the surface of the main tower and anchor room of a bridge in Dalian, Liaoning Province, China. The original images have a resolution of 4160 × 3120, but they were manually divided into smaller images of 256 × 256 resolution, classifying them into images with cracks and images without cracks. The SDNET2018 dataset provides images of concrete bridges, including both images with cracks and images without cracks [40]. The performance of artificial neural network models is related to the size of the training dataset. However, obtaining a large and diverse dataset of real crack images can be challenging. In domains where collecting a sufficient amount of real data is difficult, researchers often employ techniques such as data augmentation to compensate for the limited availability of real crack images. Data augmentation techniques can effectively address the limitations of a small training dataset such as overfitting by artificially expanding it with synthetically generated data [41].

One simple data augmentation technique is to use the ImageDataGenerator interface provided in TensorFlow 2.0, which is modeled based on spatial-level transform. By applying operations such as image flipping, rotation, shifting, and other transformations, we can effectively increase the number of data samples [42]. However, since most of them are simple linear transforms, they may not be sufficient to represent the diversity of crack patterns (see Figure 1).

## III. PROPOSED FRAMEWORK

In the proposed method, we utilize 1) a CNN for extracting crack patterns and 2) elastic distortion for transforming the shape of cracks. Since cracks do not have a standardized shape and can vary depending on the surrounding environment, it is necessary to train and learn their characteristics in order to create metadata that considers the features of real crack data. In this section, we will discuss a learning-based data augmentation method that utilizes the direction and thickness of cracks.

### A. EXTRACTING SKELETONS FROM CRACK DATA

Upon examining real crack data, it is observed that the areas where cracks exist are very small in size. Therefore, when applying data transformations to the crack line segments, the cracks tend to be irregularly fragmented (see Figure 2). To preserve the connectivity information of crack line segments, our research first extracts a skeleton from the crack images. To achieve this, we apply a skeletonization process [43] to convert the crack line segments from a pixel-based representation to a thin, vector-based representation.

### B. EXTRACTING CORNERS FROM SKELETON DATA

In this section, we describe a method for detecting corners, which are the intersection points of two edges with significant gradient changes, in crack data. In this paper, we utilize the Shi-Tomasi corner detection technique, which is an improvement of the Harris corner detection method [44]. This technique calculates the rate of change between edges and considers locations with significant bidirectional gradient changes as corners. It calculates the squared sum of pixel value differences within a mask window by moving one pixel at the center and shifting $u$ pixels in the $X$-axis and $v$ pixels in the $Y$-axis within the window. In Equation 1, $W(x, y)$ represents the weight at position $(x, y)$, $I(x, y)$ represents the pixel value at the original location, and $I(x + u, y + v)$ represents the pixel value at the shifted location.

$$E(u, v) = \sum_{x,y} W(x, y)\,[I(x + u, y + v) - I(x, y)]^2 \quad (1)$$

Applying Taylor expansion to the above equation yields Equation 2.

$$E(u, v) \cong \sum_{x,y} \left[I(x, y) + uI_x + vI_y - I(x, y)\right]^2 \quad (2)$$

Equation 3 is the final equation obtained by rearranging Equation 2 into a matrix form. If the eigenvalue of the matrix is higher than a certain threshold, it is considered as a corner. This process is calculated using the Singular Value Decomposition (SVD) technique.

$$E(u, v) \cong [u, v] \left( \sum_{x,y} W(x, y) \begin{bmatrix} I^2x & I_xI_y \\ I_xI_y & I^2y \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix} \quad (3)$$

If the user specifies the maximum number of corners, the threshold for gradient change, and the minimum distance between corners, $N$ corners will be extracted. Figure 3d shows the result of connecting the detected corners with line segments. This data will be utilized in direction-based augmentation techniques. The Figure demonstrates the stable extraction of crack skeletons without any disconnected segments, which is evident when compared to Figure 2.

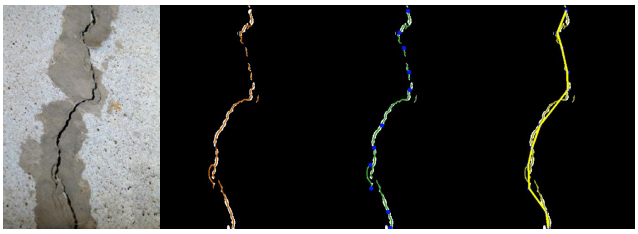



**FIGURE 3.** Crack vector with skeleton and corner set : (a) input data, (b) skeleton extraction, (c) corner extraction, (d) edge direction created by connecting corners.

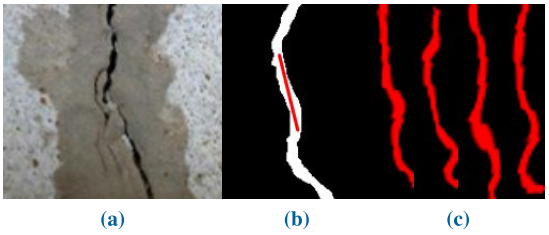



**FIGURE 4.** Direction-based elastic distortion in crack direction.

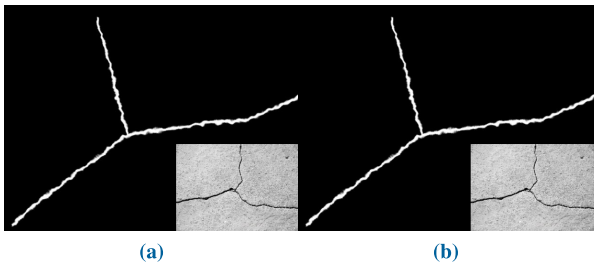



**FIGURE 5.** Crack data augmentation with our method (inset image : input data).

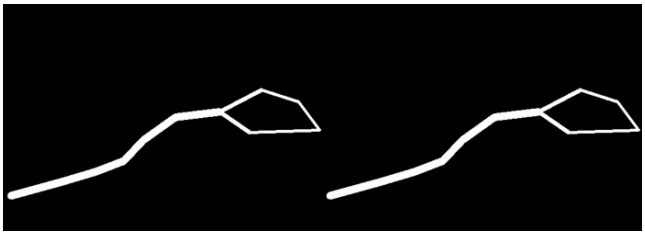



**FIGURE 6.** Crack generation with CNN based our method.

### C. DIRECTION-BASED ELASTIC DISTORTION

In this section, we describe the direction-based elastic distortion technique that we modeled from both micro and macro perspectives. At the micro level, we calculate the deformations for cracks by adjusting the weights of the standard deviation $\sigma$ and the deformation intensity $\alpha$, similar to previous studies [45]. Figure 4b is an enlarged view of one of the line segments formed by connecting the extracted corners, as mentioned earlier. As can be seen in the figure, it allows us to easily observe the extent of deformation based on the applied weights. Figure 4c shows the results of applying direction-based elastic distortion to the line segment representing cracks. From a macroscopic perspective, elastic distortion involves adjusting the rotation angle and the weights of affine transforms to deform the surface shape of the crack.

Figure 5 shows the results of applying micro-macro elastic distortion using direction-based segments. Unlike traditional pixel-based data augmentation techniques, which were difficult to control for such directionality, our approach can capture the direction of cracks with more diverse variations, rather than simple straight-line cracks. Certainly, introducing randomness can temporarily mitigate this issue. However, transformations based on randomization can lead to the loss of original crack patterns and characteristics.

### D. FEATURE-LEARNING WITH ARTIFICIAL NEURAL NETWORKS

In this section, we introduce a CNN-based network that can effectively model crack data in detail by training on the proposed direction-based data augmentation technique. For training, we utilized 500 skeletonized crack images and 400 line segment images. In addition, we collected additional training data using the direction-based data augmentation technique. In this section, we describe the process of feature learning based on the acquired crack images to obtain patterns that resemble real cracks. When training the network using the crack images obtained through the skeletonization technique, the features were complex, leading to convergence issues during the learning process. However, this problem was addressed by extracting corners from the cracks and utilizing the direction data obtained by connecting them with line segments.

Figure 6a represents the skeleton data extracted from the crack images, while Figure 6b displays the results generated through CNN training, showing patterns that closely resemble real cracks. When the weight of elastic distortion is large, it can introduce noise that may not fully preserve the characteristics of the original cracks. However, the proposed method in this paper has mitigated this issue.

The architecture of the residual-based CNN used in the experiment is shown in Figure 7. By utilizing direction-based training images, a network with 26 layers, 3 upsampling steps, and a residual image as input, we were able to achieve optimal results during the network training process. As the number of upsampling steps increased, the results improved, and by using a residual image, we addressed the color-related issues in the 3-channel training. Finally, to obtain results that consider the details of crack patterns, we used direction-augmented elastic distortion data as input and trained the network using this data to obtain the results. However, this approach only considers the direction of cracks and does

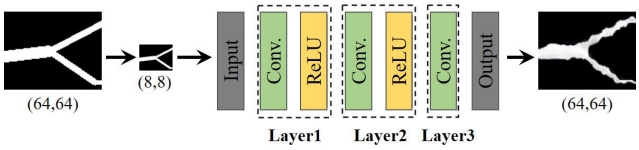



**FIGURE 7.** Our network architecture.

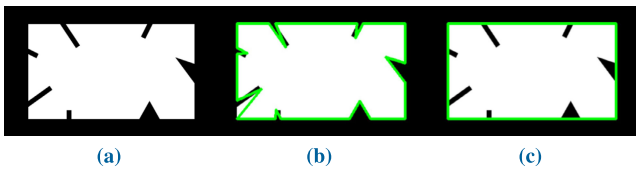



**FIGURE 8.** Contour approximation : (a) input data, (b) contour image with $\varepsilon \geq 0.005$, (c) contour image with $\varepsilon$=0.005.

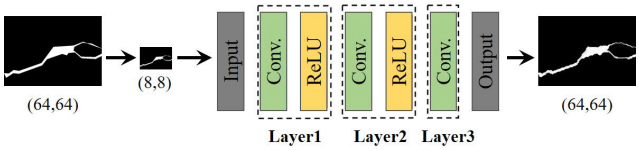



**FIGURE 9.** Advanced our network architecture.

not take into account the thickness of the cracks, so it may not be sufficient to fully capture the variations in crack thickness during training. Therefore, in this paper, we extend the framework to incorporate a method for learning crack thickness.

### *E. LEARNING THE THICKNESS OF CRACK*

In this paper, we utilize contour approximation to analyze the shape of cracks and measure their thickness. Contour refers to the boundary that is formed by regions with the same color or pixel values. It is commonly used to identify the outline or shape of an object. In this paper, the Douglas-Peucker algorithm, which is embedded in the OpenCV library, was used. To achieve results of Figure 8, the parameter $\varepsilon$ for contour approximation was set to 0.005. This value is empirically determined and commonly used to obtain tight contours. A smaller $\varepsilon$ value will yield results that closely resemble the original contour, while a larger $\varepsilon$ value may result in differences from the original contour.

The final data extracted in this study captures more detailed features in terms of thickness compared to the previous experiment results that only considered directionality. Figure 9 represents an artificial neural network trained using the thickness of cracks extracted based on contour information. The network model follows the approach described earlier, and the input data consists of both the thickness and direction information of cracks.

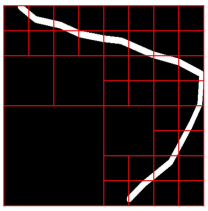


**FIGURE 10.** Quadtree construction.

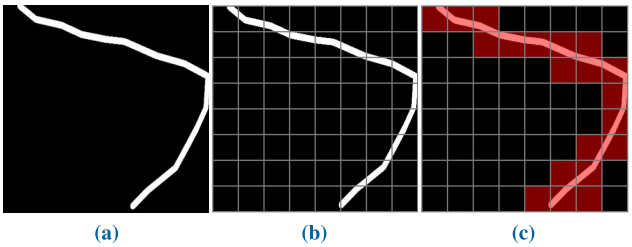


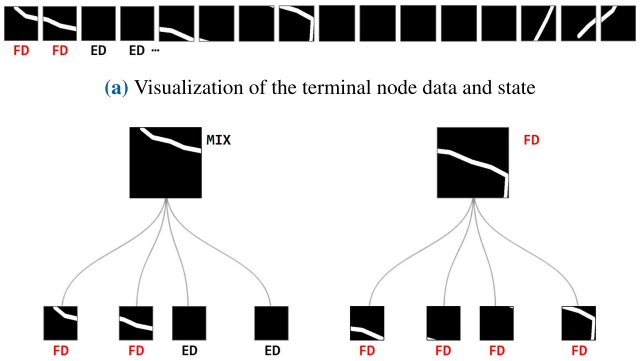

**FIGURE 11.** Classification as full density(FD) and emptry density(ED) according to the presence or absence of density : (a) input data, (b) patch split, (c) classification of FD(red) and ED(others).

## IV. SOLVER EXTENSION : ADAPTIVE OPTIMIZATION

Quadtree is a tree data structure where each internal node has four children, and it adaptively subdivides based on the presence of significant data in each node. Depending on the application, the ''minimum unit of interest'' in terms of information is typically stored in the leaf cells of a quadtree (see Figure 10). In this paper, the quadtree is used to reduce the size of the data because cracks occupy a small portion of the overall space compared to the entire dataset.

In this paper, binary data representing cracks as white areas is used as input (see Figure 11a). As shown in Figures 11b and 11c, the spatial is divided based on the crack regions as a reference for the quadtree subdivision. The reason for using a quadtree to partition the crack data is not to obtain pixel information, but rather to extract meaningful information from the spatial domain. The network training is performed using image patches represented by the leaf nodes of the quadtree, which are denoted as $C_p$. The lowest level nodes used to construct the quadtree are generated using the method described earlier, and the nodes are merged in an upward manner to form the tree structure.

Before combining the generated nodes into the quadtree, the density value of each node is compared to a threshold to determine if a crack exists within the node. Based on this comparison, the nodes are classified as either FD (Full Density) or ED (Empty Density) states (see Figure 12a). The lowest-level nodes have their own classified state values, and the state value of higher-level nodes is determined by combining the state values of their child nodes (see Figure 12b). Each node in the tree has a state value, density data, and a key value associated with it. The key contains the $X$ and $Y$ coordinates representing the node's position and tree depth used to construct the tree. The depth and position




(a) Visualization of the terminal node data and state

(b) Generating the state value of parent nodes in a bottom-up manner

**FIGURE 12. Overview of generating the quadtree containing path-state values(ED=empty density, FD=full density).**

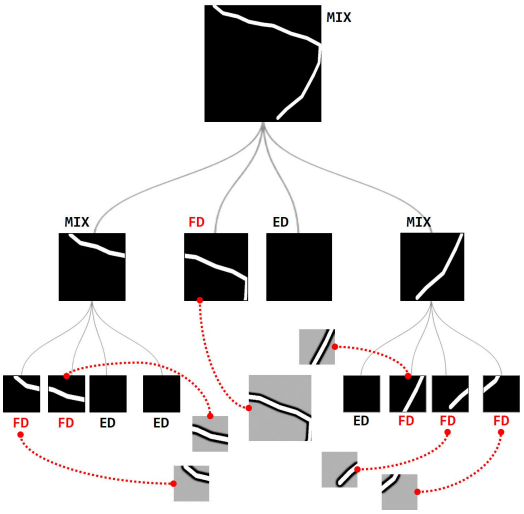



**FIGURE 13. Example of quadtree nodes for collecting crack patches (dotted red line : final dataset).**

are used after the network process is completed to combine the results. The state is represented as FD (Full Density), ED (Empty Density), or MIX. In this paper, the density of each patch is used as the data for the lowest-level nodes (see Figure 12a). The depth of the lowest-level nodes is calculated using the following equation (see Equation 4).

$$d = log_2\left(\frac{D_{width}}{N_{width}}\right) \tag{4}$$

where $d$ represents the depth of the current node, while $D_{width}$ and $N_{width}$ respectively indicate the width of the entire input data and the width of the current node. The parent node is generated in an upward manner from its four child nodes. The data of the parent node is the sum of the data from its child nodes. The depth decreases by 1, and the position is determined when merging the child nodes. Finally, the parent node's state value is determined by aggregating the states of its child nodes : If all child nodes have the same state value, that value is assigned to the parent node. If the child nodes have both FD and ED states, the state value of the parent node is assigned as MIX.

If all child nodes have the same state value, the parent node stores that value and the child nodes are removed. Nodes with the state ED indicate the absence of cracks, so they do not require network training. In the case of FD, instead of applying network training all at once, this paper follows a more efficient approach by applying it selectively to the child nodes containing the necessary data. By repeating this process until reaching the root node, a quadtree with assigned state values for all nodes is constructed. Once the tree is completed, the data and key values of the FD nodes are collected (see Figure 13), and this dataset is used for network training. The experimental results in this paper showed that overall, approximately 37.5% of the data could be compressed, leading to improved memory efficiency.

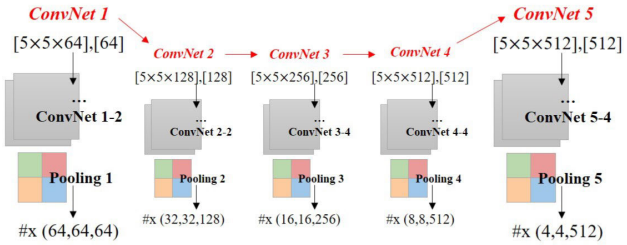



**FIGURE 14. Convolutional neural network architecture for cracks(input : x(128,128,3), output : x(4, 4, 512), [weight][bias], #x(width,height,depth)).**

## V. IMPLEMENTATION DETAILS

This paper presents a data augmentation technique based on neural networks to efficiently learn the direction and thickness of cracks. The hyperparameters used for training are as follows: The network architecture was designed with a batch size of 128, a learning rate of 1e-4, and a stride of 1. In this paper, we designed the network using the VGG19 convolutional neural network, and in this section, we will provide a detailed explanation of the VGG19 network architecture. The VGGNet model consists of two main stages, as follows : VGGG network and reconstruction. For example, when receiving a 3-channel image with a resolution of 128 × 128 as input, the input **x** can be represented as follows : **x**(128,128,3).

As mentioned earlier, the first ConvNet1 consists of two ConvNet layers. The weight and bias sizes used in this case are (5 × 5×64, 64). The output dimensions of ConvNet1 in terms of width, height, and depth are (64, 64, 64). The activation function used here is ReLU (Rectified Linear Unit). The output of ConvNet1 is used as the input for ConvNet2, and the overall VGGNet model is as shown in Figure 14. In this figure, ''ConvNet 1-2'' means that two ConvNet layers are used in the ConvNet1 process, and similarly, ''ConvNet

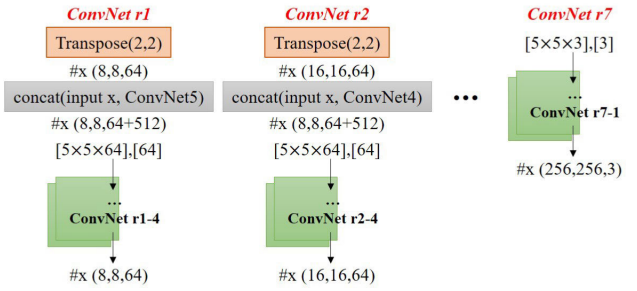



**FIGURE 15.** Feature reconstruction (input : x(4,4,512), output : x(256, 256, 3).


**FIGURE 16.** Generation of virtual cracks via data augmentation from input data (left to right : 0°, -45°, 45°, 90° ).

5-4" means that four ConvNet layers are used in the ConvNet5 process.

In the reconstruction phase, the results are restored using transposed convolution applied to the output of the VGG network (see Figure 15). The detailed information for ConvNet r1 to r7 is as shown in Table 1. The input of the neural network is half the size of the original map, and when combined with the residual map in the future, it is added at the same size as the output of the reconstruction. This network was implemented in TensorFlow [38], and the optimizer used here is Adam (Adaptive Moment Estimation).

## VI. EXPERIMENTAL RESULTS

To analyze the proposed method from various perspectives, we perform extensive augmentation of crack data and generate synthetic data. Subsequently, the generated synthetic data is applied to the network architecture designed for crack detection, and validation tests are conducted. By testing the diverse crack data with different thicknesses and orientations, we examined the impact of the synthetic data generated using our method on the training process of the network. Additionally, we examined the impact of our method on crack detection results by utilizing not only clean data but also blurred data affected by motion blur. This allowed us to assess how significantly our approach affects the crack detection accuracy in the presence of blurring effects. Lastly, we conducted experiments to investigate whether our method produces meaningful results for crack detection in glass material as well, not just concrete.

### A. CRACK DATA AUGMENTATION

To conduct more intuitive augmentation tests, we performed data augmentation experiments on input data with relatively low directional freedom in the form of straight lines, where specific angles were tested. Figure 16 displays virtual cracks generated through our method, showcasing the augmented results that align with various directions despite the input being in the form of straight lines.

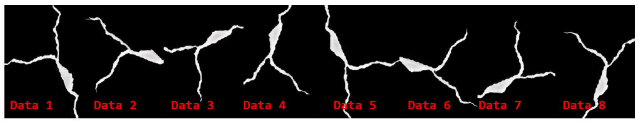
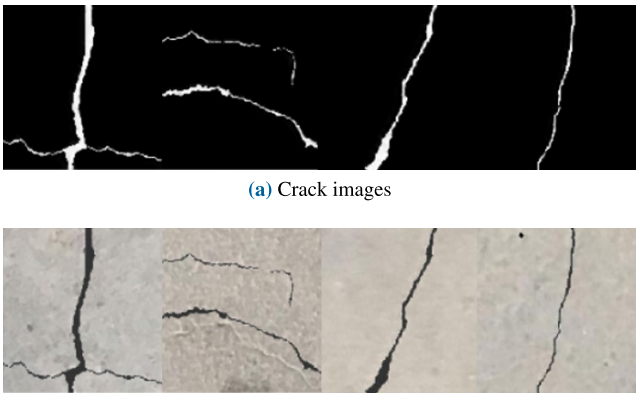



**FIGURE 17.** Crack data augmentation with our method.


**FIGURE 18.** Synthesizing crack images and masks to generate synthetic data : (a) crack image, (b) concrete material DB, (c) synthetic data.



(a) Crack images

(b) Synthesized cracks

**FIGURE 19.** Comparison of synthesized cracks and their masks.

Figure 17 shows the results of generating cracks with diverse patterns using the data augmentation technique proposed in this paper. Unlike previous data augmentation approaches that performed elastic distortion at the pixel level, the proposed method enables easy generation of complex crack datasets considering various directions and thicknesses. To utilize this dataset for crack detection, it is necessary to establish pairs of mask images and crack images. We pre-built a database that can represent various concrete materials and synthesized synthetic data by combining it with crack data (see Figure 18). Figure 19 shows the synthetic data generated by synthesizing virtual crack images and concrete materials in this paper. The neural network training for crack detection is conducted using pairs of images that are represented in a similar form to real crack images.

**TABLE 1.** Details of the crack convolutional neural network architecture.

| | ConvNet r1 | ConvNet r2 | ConvNet r3 | ConvNet r4 | ConvNet r5 | ConvNet r6 | ConvNet r7 |
|---|---|---|---|---|---|---|---|
| Transpose(...) | (2,2) | (2,2) | (2,2) | (2,2) | (2,2) | (2,2) | – |
| #x(w,h,d) | (8,8,64) | (16,16,64) | (32,32,64) | (64,64,64) | (128,128,64) | (256,256,64) | – |
| concat(...) | (input x, ConvNet5) | (input x, ConvNet4) | (input x, ConvNet3) | (input x, ConvNet2) | (input x, ConvNet1) | – | – |
| #x(w,h,d) | (8,8,64+512) | – | – | – | – | – | – |
| [weight],[bias] | 5×5×64,64 | 5×5×64,64 | 5×5×64,64 | 5×5×64,64 | 5×5×64,64 | 5×5×64,64 | 5×5×3,3 |
| Num. ConvNet | ConvNet r1-4 | ConvNet r2-4 | ConvNet r3-4 | ConvNet r4-2 | ConvNet r5-2 | ConvNet r6-2 | ConvNet r7-1 |
| #x(w,h,d) | (8,8,64) | (16,16,64) | (32,32,64) | (64,64,64) | (128,128,64) | (256,256,64) | (256,256,3) |

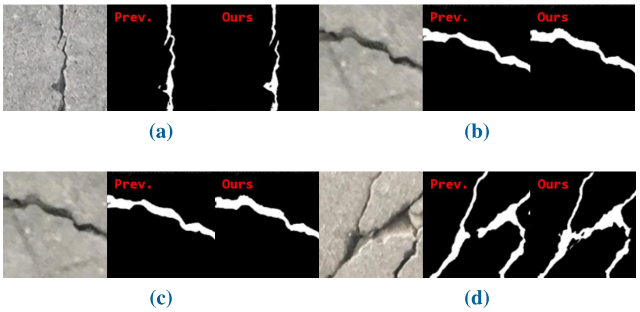



**FIGURE 20.** Quality comparison between previous method [39] and our method (scene 1). In each subfigure, the left, middle, and right figures are the input data, the previous method, and our method.

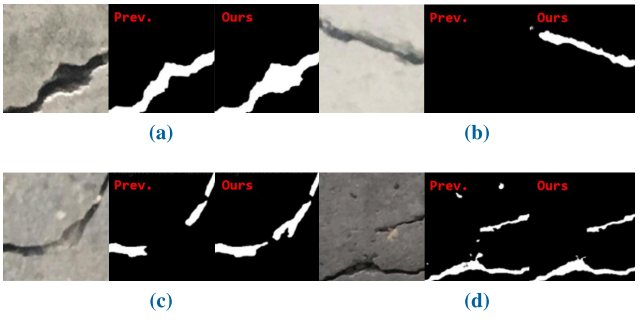



**FIGURE 21.** Quality comparison between previous method [39] and our method (scene 2). In each subfigure, the left, middle, and right figures are the input data, the previous method, and our method.

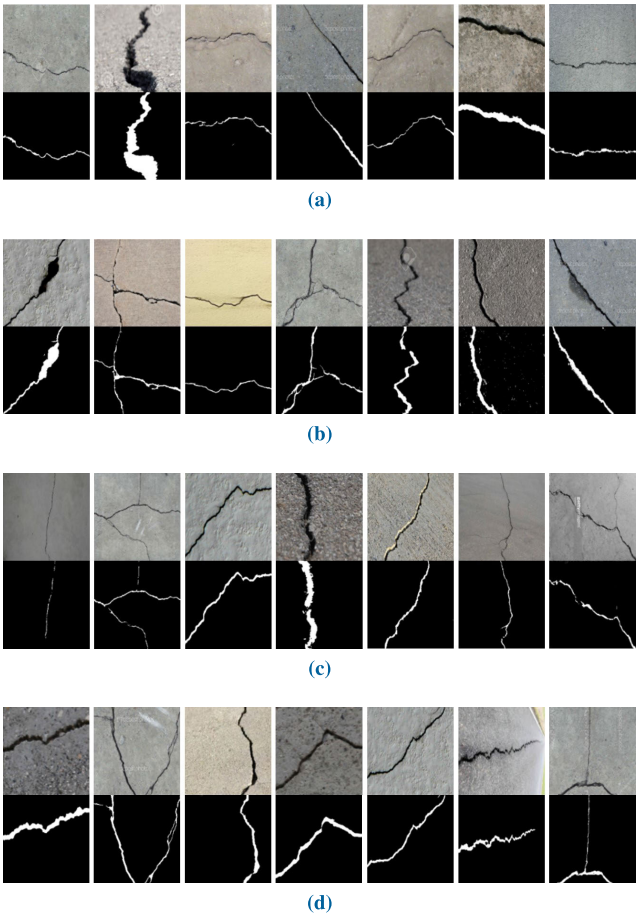



**FIGURE 22.** Results of extracted crack area with our method (scene 3).

### B. CRACK DETECTION

In this section, we apply the proposed data augmentation technique to crack detection and examine the results. In this study, we augmented the data provided by DeepCrack to create a total of 3,000 images [39]. All the data was labeled with masks according to the presence or absence of cracks, and it was divided into training, testing, and validation sets with a ratio of 6:2:2.

Figure 20 presents the results of crack detection using our method. Compared to the input data, the previous method failed to accurately represent the details of the cracks, resulting in discontinuities or introducing noise. In contrast, our method, with the addition of data augmentation, was able to detect relatively clean cracks.

Figure 21 shows the results of comparing our method with the previous method using a variety of crack patterns as input. In Figure 21a, it can be observed that the previous method fails to accurately detect the crack thickness that is present in the input data. However, our method demonstrates more accurate detection of the thick cracks. Figure 21b presents the results of the experiments conducted on cracks observed in relatively lighter-colored concrete. While the previous method struggled to detect the crack, our method, with the use of data augmentation alone, successfully identified the entire crack. In Figure 21c, it can be observed that the previous

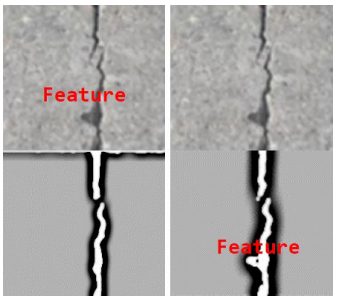



**FIGURE 23.** Experimental results with shaky crack data. This illustration shows contour filtering applied to clearly see the results : (a) previous method [39], (b) our method.

method detected the cracks in a discontinuous manner, while our method provided more stable and consistent results. In Figure 21d, not only cracks but also two small holes can be observed at the top of the input image. Such holes are typically not classified as cracks, but the previous method mistakenly detected them as cracks. In contrast, our method accurately detects only the actual crack portions while disregarding the holes.

Figure 22 presents the results of extracting crack areas from the input image using our method. It demonstrates stable crack detection across various crack patterns. The proposed method, being a data augmentation technique, is expected to be applicable as an add-on to various approaches that utilize crack data, rather than being limited to a specific neural network.

Additionally, experiments are conducted to examine whether the proposed method can effectively extract cracks from blurred images caused by motion blur. Cracks often occur in locations where it is challenging for users to maintain a stable posture, such as high areas. In such cases, the photos may exhibit blur due to shake caused by the unstable posture of the user. In this experiment, we aim to assess how effectively our method can detect cracks in cases where there is shake-induced blur (see Figure 23). The input data at the top of Figure 23 represents a photo with blur caused by shake. In Figure 23a, we can observe the crack detection results using the previous method that does not utilize data augmentation. On the other hand, Figure 23b shows the results of crack detection after applying data augmentation. The results of crack detection using data augmentation demonstrate a more accurate representation of crack shapes (see Figure 23b).

Figure 24 shows the results of crack detection performed on shattered glass. Similar to the previous results, the crack detection results after data augmentation (see Figure 24c) are more sharp and accurate compared to the results before data augmentation (see Figure 24b).

### C. CRACK DETECTION NETWORK

In this paper, DeepCrack was chosen as the neural network for crack detection [39]. We generate crack data and create pairs

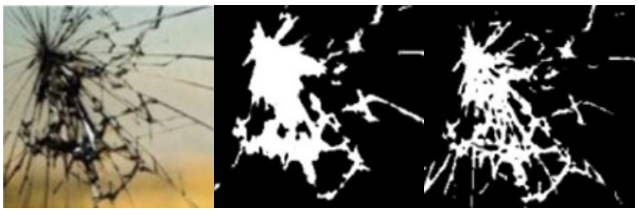



**FIGURE 24.** Experimental results with glass crack data : (a) input data, (b) previous method [39], (c) our method.

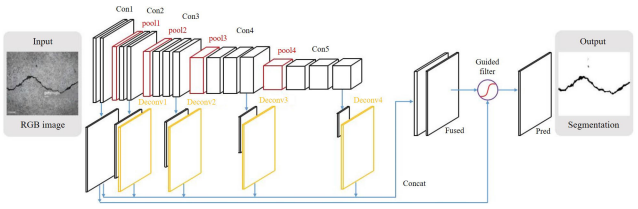



**FIGURE 25.** An illustration of our proposed DeepCrack architecture [39].

by synthesizing them with concrete material images. These pairs are then used for training the neural network. During the test phase, we input real concrete images and perform crack detection to identify the regions of cracks. I recommend reading the DeepCrack paper for a more detailed explanation of the methodology [39].

### D. EVALUATION

In this study, Mean IoU (Intersection over Union) was used as the evaluation metric. This method evaluates the model by measuring the intersection over union ratio between the predicted results and the ground truth values in deep learning. Generally, an IoU value of 0.5 or higher is considered as a successful detection (True Positive, TP), while a value below 0.5 is regarded as a false detection (False Positive, FP). It was observed that without data augmentation, the false positive (FP) rate was approximately 25%. However, when applying the proposed method, which involved augmenting the crack data more than tenfold, the FP rate decreased to as low as 3%. Additionally, the accuracy of IoU improved from 0.5803 to 0.9223. In this experiment, the initial number of crack images was 237, and through augmentation, a total of 1,659 images were used.

## VII. CONCLUSION

The paper proposed a framework that augmented crack data based on network training, enabling the generation of diverse patterns of crack data by learning the direction and thickness of cracks in detail. Furthermore, the experiments demonstrated that the use of augmented data improved the accuracy of real crack detection. The proposed method was applied to cracks represented in concrete material, but it is expected to be extended to various types of material cracks as well.

However, this study has several limitations. It is anticipated that detecting cracks in materials such as glass, ceramics, plastic, and aluminum, which involve significant reflection and refraction, may be challenging as the algorithm was designed and tested specifically for concrete materials. Additionally, since the synthetic data generation assumes dark-colored cracks, it may be challenging to detect bright-colored cracks or scratches. My future plan is to research methods for automatically detecting and evaluating the severity of cracks, taking into account the characteristics of various materials.

## REFERENCES

[1] X. Wang, P. Liu, Y. Sun, and W. Wang, "Study on breaking concrete structures by pulse power technology," *Buildings*, vol. 12, no. 3, p. 274, Feb. 2022.

[2] D. G. Aggelis, N. Alver, and H. K. Chai, "Health monitoring of civil infrastructure and materials," *Sci. World J.*, vol. 2014, Feb. 2014, Art. no. 435238.

[3] I.-H. Kim, H. Jeon, S.-C. Baek, W.-H. Hong, and H.-J. Jung, "Application of crack identification techniques for an aging concrete bridge inspection using an unmanned aerial vehicle," *Sensors*, vol. 18, no. 6, p. 1881, Jun. 2018.

[4] T. Liu, H. Huang, and Y. Yang, "Crack detection of reinforced concrete member using Rayleigh-based distributed optic fiber strain sensing system," *Adv. Civil Eng.*, vol. 2020, pp. 1–11, Jul. 2020.

[5] T. Yamaguchi, S. Nakamura, R. Saegusa, and S. Hashimoto, "Image-based crack detection for real concrete surfaces," *IEEJ Trans. Electr. Electron. Eng.*, vol. 3, no. 1, pp. 128–135, Jan. 2008.

[6] Y.-C. Tsai, V. Kaul, and R. M. Mersereau, "Critical assessment of pavement distress segmentation methods," *J. Transp. Eng.*, vol. 136, no. 1, pp. 11–19, Jan. 2010.

[7] D. Zhang, Q. Li, Y. Chen, M. Cao, L. He, and B. Zhang, "An efficient and reliable coarse-to-fine approach for asphalt pavement crack detection," *Image Vis. Comput.*, vol. 57, pp. 130–146, Jan. 2017.

[8] A. Ayenu-Prah and N. Attoh-Okine, "Evaluating pavement cracks with bidimensional empirical mode decomposition," *EURASIP J. Adv. Signal Process.*, vol. 2008, no. 1, pp. 1–7, Dec. 2008.

[9] P. Subirats, J. Dumoulin, V. Legeay, and D. Barba, "Automation of pavement surface crack detection using the continuous wavelet transform," in *Proc. Int. Conf. Image Process.*, Oct. 2006, pp. 3037–3040.

[10] L. Ying and E. Salari, "Beamlet transform-based technique for pavement crack detection and classification," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 25, no. 8, pp. 572–580, Nov. 2010.

[11] A. Hizukuri and T. Nagata, "Development of a classification method for a crack on a pavement surface images using machine learning," in *Proc. 13th Int. Conf. Quality Control Artif. Vis.*, vol. 10338, 2017, pp. 148–154.

[12] P. P. Acharjya, R. Das, and D. Ghoshal, "Study and comparison of different edge detectors for image segmentation," *Global J. Comput. Sci. Technol.*, vol. 12, no. 13, pp. 29–32, 2012.

[13] A. Graps, "An introduction to wavelets," *IEEE Comput. Sci. Eng.*, vol. 2, no. 2, pp. 50–61, Jun. 1995.

[14] L. Zhang, F. Yang, Y. Daniel Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3708–3712.

[15] L. Pauly, H. Peel, S. Luo, D. Hogg, and R. Fuentes, "Deeper networks for pavement crack detection," in *Proc. Int. Symp. Autom. Robot. Construction (IAARC)*, Jul. 2017, pp. 479–485.

[16] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, and H. Omata, "Road damage detection using deep neural networks with images captured through a smartphone," 2018, *arXiv:1801.09454*.

[17] H. Xu, X. Su, Y. Wang, H. Cai, K. Cui, and X. Chen, "Automatic bridge crack detection using a convolutional neural network," *Appl. Sci.*, vol. 9, no. 14, p. 2867, Jul. 2019.

[18] Y. Li, Z. Han, H. Xu, L. Liu, X. Li, and K. Zhang, "YOLOv3-lite: A lightweight crack detection network for aircraft structure based on depthwise separable convolutions," *Appl. Sci.*, vol. 9, no. 18, p. 3781, Sep. 2019.

[19] Z. Tong, J. Gao, and H. Zhang, "Recognition, location, measurement, and 3D reconstruction of concealed cracks using convolutional neural networks," *Construction Building Mater.*, vol. 146, pp. 775–787, Aug. 2017.

[20] X. Yang, H. Li, Y. Yu, X. Luo, T. Huang, and X. Yang, "Automatic pixel-level crack detection and measurement using fully convolutional network," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 33, no. 12, pp. 1090–1109, Dec. 2018.

[21] J. Zhu and J. Song, "An intelligent classification model for surface defects on cement concrete bridges," *Appl. Sci.*, vol. 10, no. 3, p. 972, Feb. 2020.

[22] S. Ren, K. He, R. Girshick, and S. Jian, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.

[23] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Jul. 2016, pp. 1–9.

[24] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2117–2125.

[25] L. Deng, H.-H. Chu, P. Shi, W. Wang, and X. Kong, "Region-based CNN method with deformable modules for visually classifying concrete cracks," *Appl. Sci.*, vol. 10, no. 7, p. 2528, Apr. 2020.

[26] H. Oh, N. W. Garrick, and L. E. K. Achenie, "Segmentation algorithm using iterative clipping for processing noisy pavement images," in *Proc. Imaging Technol., Techn. Appl. Civil Eng. 2nd Int. Conf.*, 1998, pp. 138–147.

[27] Q. Li and X. Liu, "Novel approach to pavement image segmentation based on neighboring difference histogram method," in *Proc. Congr. Image Signal Process.*, vol. 2, May 2008, pp. 792–796.

[28] J. Zhou, "Wavelet-based pavement distress detection and evaluation," *Opt. Eng.*, vol. 45, no. 2, Feb. 2006, Art. no. 027007.

[29] S. Wu and Y. Liu, "A segment algorithm for crack dection," in *Proc. IEEE Symp. Electr. Electron. Eng. (EEESYM)*, Jun. 2012, pp. 674–677.

[30] Q. Li, Q. Zou, D. Zhang, and Q. Mao, "FoSA: F* seed-growing approach for crack-line detection from pavement images," *Imag. Vis. Comput.*, vol. 29, no. 12, pp. 861–872, Nov. 2011.

[31] Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang, "CrackTree: Automatic crack detection from pavement images," *Pattern Recognit. Lett.*, vol. 33, no. 3, pp. 227–238, Feb. 2012.

[32] F. Roli, "Measure of texture anisotropy for crack detection on textured surfaces," *Electron. Lett.*, vol. 32, no. 14, pp. 1274–1275, Jul. 1996.

[33] T. S. Nguyen, S. Begot, F. Duculty, and M. Avila, "Free-form anisotropy: A new method for crack detection on pavement surface images," in *Proc. 18th IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 1069–1072.

[34] W. Xu, Z. Tang, J. Zhou, and J. Ding, "Pavement crack detection based on saliency and statistical features," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2013, pp. 4093–4097.

[35] Y. Hu and C.-X. Zhao, "A novel LBP based methods for pavement crack detection," *J. Pattern Recognit. Res.*, vol. 5, no. 1, pp. 140–147, 2010.

[36] W. Zhang, Z. Zhang, D. Qi, and Y. Liu, "Automatic crack detection and classification method for subway tunnel safety monitoring," *Sensors*, vol. 14, no. 10, pp. 19307–19328, Oct. 2014.

[37] S. Li and X. Zhao, "Image-based concrete crack detection using convolutional neural network and exhaustive search technique," *Adv. Civil Eng.*, vol. 2019, pp. 1–12, Apr. 2019.

[38] M. Abadi, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*.

[39] Y. Liu, J. Yao, X. Lu, R. Xie, and L. Li, "DeepCrack: A deep hierarchical feature learning architecture for crack segmentation," *Neurocomputing*, vol. 338, pp. 139–153, Apr. 2019.

[40] S. Dorafshan, R. J. Thomas, and M. Maguire, "SDNET2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks," *Data Brief*, vol. 21, pp. 1664–1668, Dec. 2018.

[41] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," 2017, *arXiv:1712.04621*.

[42] P. Singh, A. Manure, P. Singh, and A. Manure, "Introduction to tensorflow 2.0," in *Learn TensorFlow 2.0: Implement Machine Learning and Deep Learning Models With Python*, 2020, pp. 1–24.

[43] P. K. Saha, G. Borgefors, and G. S. di Baja, "Skeletonization and its applications—A review," in *Skeletonization*, 2017, pp. 3–42.
[44] C. S. Kenney, M. Zuliani, and B. S. Manjunath, "An axiomatic approach to corner detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jul. 2005, pp. 191–197.
[45] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proc. ICDAR*, vol. 3, 2003, pp. 958–963.
[46] G. Kang, X. Dong, L. Zheng, and Y. Yang, "PatchShuffle regularization," 2017, *arXiv:1707.07103*.
[47] H. Inoue, "Data augmentation by pairing samples for images classification," 2018, *arXiv:1801.02929*.
[48] C. Summers and M. J. Dinneen, "Improved mixed-example data augmentation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 1262–1270.
[49] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," 2017, *arXiv:1710.09412*.
[50] R. Takahashi, T. Matsubara, and K. Uehara, "Data augmentation using random image cropping and patching for deep CNNs," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 9, pp. 2917–2931, Sep. 2020.
[51] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio, "Manifold mixup: Better representations by interpolating hidden states," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6438–6447.
[52] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 7, 2020, pp. 13001–13008.
[53] K. K. Singh, H. Yu, A. Sarmasi, G. Pradeep, and Y. J. Lee, "Hide-and-seek: A data augmentation technique for weakly-supervised localization and beyond," 2018, *arXiv:1811.02545*.
[54] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, and J. Choe, "CutMix: Regularization strategy to train strong classifiers with localizable features," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6023–6032.
[55] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, "AugMix: A simple data processing method to improve robustness and uncertainty," 2019, *arXiv:1912.02781*.
[56] J.-H. Lee, M. Z. Zaheer, M. Astrid, and S.-I. Lee, "SmoothMix: A simple yet effective data augmentation to train robust classifiers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 756–757.
[57] J.-H. Kim, W. Choo, and H. O. Song, "Puzzle mix: Exploiting saliency and local statistics for optimal mixup," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5275–5285.