Chapter

Computer Vision-Based Techniques for Quality Inspection of Concrete Building Structures

Siwei Chang and Ming-Fung Francis Siu

Abstract

Quality performance of building construction is frequently assessed throughout the construction life cycle. In Hong Kong, quality management system must be established before commencing new building works. Regular building inspections are conducted in accordance with the code of practice of new building works. Quality managers are deployed in construction sites to inspect and record any building defects. The concrete cracks must be identified, which is usually followed by proposed rectifications, in order to protect the public and occupants from dangers. This chapter is structured as follows: Background information of concrete cracks is firstly given. Traditional technique of conducting regular manual inspection is introduced, in accordance with Hong Kong's code of practice "Building Performance Assessment Scoring System (PASS)". Then, an advanced technique of conducting crack inspection intelligently based on computer vision is introduced. The procedures of defining, training, and benchmarking the architecture of convolutional neural network models are presented. The calculation steps are detailed and illustrated using a simple text-book example. An experiment case study is used to compare the time, cost of inspecting concrete cracks using both manual and advanced technique. The study concludes with a presentation of the future vision of robot-human collaboration for inspecting concrete cracks in building construction.

Keywords: building quality control, concrete crack, quality inspection, computer vision, artificial intelligence

1. Introduction

Throughout the entire construction life cycle, quality assessment plays an important role in ensuring the safety, economy, and long-term viability of construction activities. Construction products that have been completely inspected and certificated by quality inspectors are more inclined to be chosen by developers and buyers. Typically, the structural work is considered as an essential aspect for quality assessment because structural problems directly influence the construction stability and integrity. Among the construction structural forms, concrete structures are adopted as the most common and basic construction structure. Therefore, exploring advanced technologies that enable effective concrete defect inspection can be deemed a worthwhile endeavor.

Normally, the types of concrete defects include blistering, delamination, dusting, etc. Among them, concrete cracks, usually caused by deformation, shrinkage, swelling, or hydraulic, appear most frequently in concrete components. Concrete cracking is considered the first sign of deterioration. As reported by the BRE Group [1], cracks up to 5 mm in width simply need to be re-decorated because they only affect the appearance of the concrete. However, cracks with a width of 5–25 mm have the possibility to trigger structural damage to concrete structures [2]. A 40-year-old oceanfront condo building collapsed on June 27, 2021, in Florida because of the neglect of cracks. Experienced engineers noticed the cracked or crumbling concrete, the interior cracks, and the cracks at the corners of windows and doors are the significant and earliest signs of this tragedy. Therefore, in order to prevent potential failures that may pose a loss to society, crack problems should be thoroughly examined and resolved.

In general, construction works are divided into two categories: new building works and existing building works. The new works refer to a building that will be constructed from scratch. The existing building works mean that a building has existed for many years and residents are living inside. In Hong Kong, quality assurance and control should be conducted by full-time quality managers on-site for both new and existing buildings. Normally, the quality managers visually inspect implied build quality and by appointing a score to the building's quality in accordance to the Building Performance Assessment Scoring System (PASS) for new buildings, the Mandatory Building Inspection Scheme (MBIS), and the Mandatory Window Inspection Scheme (MWIS) for existing buildings. Meanwhile, to ensure a continuous and in-depth inspection, Non-destructive (NDT) methods e.g., eddy current testing, ultrasonic testing are also commonly applied in the quality inspection process.

Quality managers are commonly obliged to work 8 hours per day. Their salary ranges from HKD 30,000 to HKD 50,000 per month. In PASS, more than 300 quality assessment items are related to cracking-related problems. Cracks in all building components, including floors, internal and external walls, ceilings, and others are required to be strictly inspected during both structural and architecture engineering stages. Therefore, both manual and NDT inspections are considered time-consuming, costly, and dangerous, especially for large-scale and high-rise structures. To tackle this issue, computer-vision technique is increasingly introduced for automated crack inspection. For example, various convolutional neural network (CNN) architectures have been developed and implemented to increase the efficiency of manual crack inspection [3, 4].

Considering the aforementioned context, computer-vision-based automated crack inspection techniques were introduced by the authors in 2022. To achieve this, the theoretical background of CNN networks is firstly explained in the context of convolution, pooling, fully-connected, and benchmarking processes. AlexNet and VGG16 models were then implemented and tested to detail and illustrate the calculation steps. Meanwhile, a practical case study is used to compare the difference between manual and computer-vision-based crack inspection. The future directions of combining robotics and computer-vision for automated crack inspection are discussed. This study gives a comprehensive overview and solid foundation for a computer-vision-based automated crack inspection technique that contributes to high efficiency, cost-effectiveness, and low-risk quality assessment of buildings.

2. Computer vision-based automated concrete crack inspection

The term *computer vision* is defined as an interdisciplinary field that enables computers to recognize and interpret environments from digital images or videos [5]. Computer vision techniques are rapidly being used to detect, locate, and quantify concrete defects to reduce the limitations of manual visual inspection. By automatically processing images and videos, computer vision-based defect detection technologies enable efficient, accurate, and low-cost concrete quality inspection. Various techniques in the computer vision field, such as semantic segmentation and object detection, have been developed and applied to date [6]. Among them, image classification is considered the most basic computer vision technique and has been introduced most frequently to predict and target concrete defects.

The motivation of image classification is to identify the categories of input images. Different from human recognition, an image is first presented as a three-dimensional array of numbers to a computer. The value of each number ranges from 0 (black) to 255 (white). An example is shown in **Figure 1**. The crack image is 256 pixels wide, 256 pixels tall, and has three color channels RGB (Red, Green, and Blue). Therefore, this image generates 256 × 256 × 3 = 196,608 input numbers.

The input array is then computed using computer vision algorithms to transform the numbers to a specific label that belongs to an assigned set of categories. One of the computer vision algorithms is CNN, which has become dominant in image classification tasks [7]. CNN is a form of a deep learning model for computing grid-shaped data. The central idea of CNN is to identify the image classification by capturing its features using filters. The features are then output to a specific classification by a trained weight and biases matrix.

There are three main modules included in a CNN model: convolution, pooling, and fully connected layer. The convolution and pooling layers are used to extract image features. The fully connected layer is used to determine the weight and biases matrix and to map the extracted features into specific labels.

Convolution layer is the first processing block in CNN. During the convolution process, a set of convolution filters is used to compute the input array $A = (a_{ij})_{m \times n}$, $m, n \in \left(width_{image}, height_{image}\right)$. After computing, a new image $A^* = (a^*{}_{ij})_{n \times n}$, is

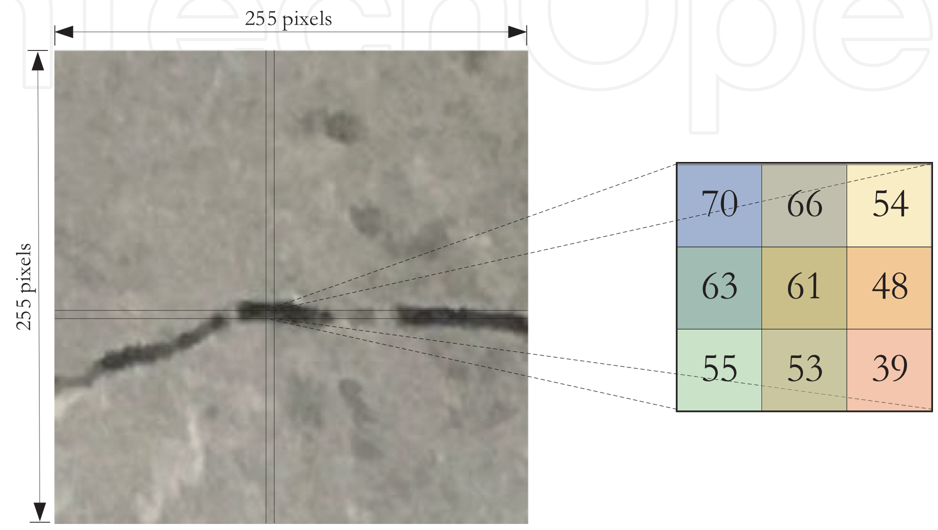

Figure 1.
An example of the input number array.

output and passed to the next processing layers. The size of the output image can be calculated with Eq. (1). The values of output image pixels can be calculated with Eq. (2). The output images are known as convolution feature map.

$$n = ((m - f + 2p)/s) + 1 \tag{1}$$

Here: n refers to the size of output image, m refers to the size of input image, f refers to the size of convolution filter, p refers to the number of pooling layer, s refers to the stride of convolution filter.

$$\mathrm{A}_o^* = f\left(\sum_k W_o \times \mathrm{A}_o + b_o\right) \tag{2}$$

Here: A_o^* refers to the pixels of output image, f refers to an applied non-linear function, W_o refers to the values of convolution filter matrix, k refers to the number of convolution filters. A_o refers to the pixels of input image, and b_o is an arbitrary real number.

An example of a convolution process is shown in **Figure 2**. In this example, both the width and height of the input image is 5. The pixels of the image are shown in **Figure 2**. The convolution filter is in a shape of 3×3. In this example, only one filter is used. The initial value of the convolution filter is set randomly. The filter matrix is adjusted and optimized in the following backpropagation process. In this example, the non-linear function, padding layer is not used, and the biases value b_o is set as 0. The stride of convolution filter is set as 1. The convolution filter moves from left to right, and from top to bottom. The size and value of the output feature map can be computed using Eqs. (1) and (2). The detailed calculation process of the example feature maps value and size is shown in **Table 1**. Seen from **Figure 2**, the value of size of input image, size of filter is 5, 3, respectively. Suppose the number of the pooling layer, the convolution stride is 0, 1, respectively.

A pooling layer is used to refine the feature maps. After pooling, the dimensions of the feature maps can be simplified. In doing so, the computation cost can be effectively decreased by reducing the number of learning parameters, whilst allowing only the essential information of feature maps to be presented. Usually, pooling layers follow behind convolution layers. Average pooling and maximum pooling are the main pooling operations. Similar to convolution layers, pooling filters are used to refine feature maps. For maximum pooling, the maximum value from the regions in feature map that is covered by pooling filters is extracted. For average pooling, the average value of the regions in feature maps covered by pooling filters is computed. The pooling filters slide in the feature map from top to bottom, and from left to right. The output of the pooling process is new feature maps that contain the most

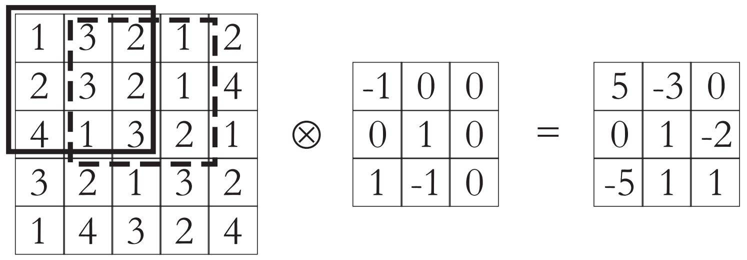

Figure 2.
An example of convolution process.

Variable	Equation	Calculation process
Size of feature map	$n = ((m - f + 2p)/s) + 1$	((5–3 + 2 × 0)/1) +1 = 3
Value of feature map	$A_o^* = f\left(\sum_k W_o \times A_o + b_o\right)$	(−1) × 1 + 0 × 3 + 0 × 2 + 0 × 2 + 1 × 3 + 0 × 2 + 1 × 4 + (−1) × 1 + 0 × 3 = 5
		(−1) × 3 + 0 × 2 + 0 × 1 + 0 × 3 + 1 × 2 + 0 × 1 + 1 × 1 + (−1) × 3 + 0 × 2 = −3
		(−1) × 2 + 0 × 1 + 0 × 2 + 0 × 2 + 1 × 1 + 0 × 4 + 1 × 3 + (−1) × 2 + 0 × 1 = 0
		(−1) × 2 + 0 × 3 + 0 × 2 + 0 × 4 + 1 × 1 + 0 × 3 + 1 × 3 + (−1) × 2 + 0 × 1 = 0
		(−1) × 3 + 0 × 2 + 0 × 1 + 0 × 1 + 1 × 3 + 0 × 2 + 1 × 2 + (−1) × 1 + 0 × 3 = 1
		(−1) × 2 + 0 × 1 + 0 × 4 + 0 × 3 + 1 × 2 + 0 × 1 + 1 × 1 + (−1) × 3 + 0 × 2 = −2
		(−1) × 4 + 0 × 1 + 0 × 3 + 0 × 3 + 1 × 2 + 0 × 1 + 1 × 1 + (−1) × 4 + 0 × 3 = −5
		(−1) × 1 + 0 × 3 + 0 × 2 + 0 × 2 + 1 × 1 + 0 × 3 + 1 × 4 + (−1) × 3 + 0 × 2 = 1
		(−1) × 3 + 0 × 2 + 0 × 1 + 0 × 1 + 1 × 3 + 0 × 2 + 1 × 3 + (−1) × 2 + 0 × 4 = 1

Table 1.
Detailed calculation process of feature map value and size.

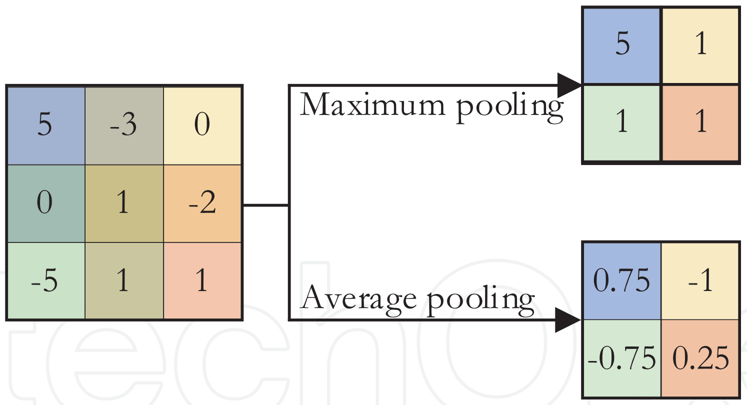

Figure 3.
An example of max pooling and average pooling.

prominent features or average features. An example of maximum pooling and average pooling is shown in **Figure 3**.

After extracting image features, the fully connected layers are applied to map these features with classification labels. The relationship between input feature maps and output classifications is calculated using an artificial neural network (ANN). The ANN is structured into input layers, hidden layers, and output layers. A group of neurons is included in the three layers. The neurons connect to one another in a processed weight matrix. The weights present the importance of input feature maps to classification labels. Therefore, the relationships between inputs and outputs can be obtained by calculating a weight matrix that connects image feature neurons and classification neurons.

To achieve this, the cube-shaped feature maps are first flattened into one-dimension vectors. The values of transformed vectors represent the values of input neurons. Then Eq. (3) is applied to calculate the value of new neurons that connect with input neurons. The initial weights and biases values are chosen at random.

$$y_j(x) = f\left(\sum_{i=1}^{n} w_j x_i + b\right) \tag{3}$$

Here: y_j refers to the weights of output neurons, w_j refers to the weights that connect different neurons, x_i refers to the values of input neurons, b refers to the biases.

A Back-Propagation algorithm (BP) is commonly used to train and modify weights and biases. BP updates weights and biases by computing the gradient of loss function. In doing this, the optimal weights and biases matrix that enable the minimum loss between model outputs and actual value are identified. For now, various loss functions are developed and applied. For example, the mean square error (MSE), shown in Eq. (4), is one of the most frequently used loss functions to calculate loss value. Stochastic gradient descent (SGD) is then processed to determine updated weights and biases using the gradient of loss function, shown as Eq. (5).

$$Loss = \frac{1}{n}\sum_{i=1}^{n} (y_{\mathrm{i}} - \hat{y}_{\mathrm{i}})^2 \tag{4}$$

Here: *Loss* refers to the loss value of output neuron and actual value, n refers to the number of neurons that connect to one specific output neuron, y refers to the actual value, $\hat{y}$ refers to the value of one output neuron.

$$w' = w - \eta \frac{\partial L}{\partial w} \tag{5}$$
$$b' = b - \eta \frac{\partial L}{\partial b}$$

Here: w', b' refers to updated weights and biases, η, η refers to former weights and biases, η refers to the learning rate, $\frac{\partial L}{\partial w}$, $\frac{\partial L}{\partial b}$ refers to the partial score of the loss function for weights and biases, respectively.

An example of feature map updating using BP is explained. **Figure 4** depicts an example of a fully connected process. The initial weights and biases in this process are determined randomly. Suppose the value of w_{11}, w_{12}, w_{21}, w_{22}, w_5, w_6 is 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, respectively. The value of x_1, x_2, actual output value is 5, 1, 0.24. The detailed calculation of the updated weights, biases, feature map is shown in **Table 2**.

In conclusion, during the convolution and pooling processes in CNN, the features of the input image are extracted first. The pooled feature maps are then flattened and considered as input neurons in fully connected process. After several training periods, the appropriate weights and biases can be determined using BP. The classifications of input images can be predicted automatically and reliably using the optimal weights and biases.

A confusion matrix is a table structure that permits the viewing of CNN performance [8]. Each row of the matrix records the number of images from actual classes, while each column records the number of images from predicted classes. There are four type indicators in the matrix: (1) True positive (TP) represents the images that are predicted correctly as the actual class; (2) False positive (FP) represents the images that are wrongly predicted; (3) True negative (TN) represents the images that are correctly predicted as another actual class; (4) False negative (FN) represents the images that are wrongly predicted as another actual class. TP, FP, TN, FN can be expressed in a 2×2 confusion matrix, shown in **Figure 5**.

Based on TP, FP, FN, and TN, four typical CNN performance evaluation indexes: accuracy, precision, recall, and F1-score can be calculated using Eqs. (6)–(9). For the crack inspection problem, accuracy shows how many images can be predicted correctly. The percentage of actual cracked photos to all predicted cracked images is shown by precision. CNNs with a high precision score indicate a better inspection ability of cracked images. Recall shows the ratio of predicted cracked images to all actual cracked images. CNNs with a high recall score indicate a better distinguishing capacity between cracked and uncracked images. F1-score shows the comprehensive

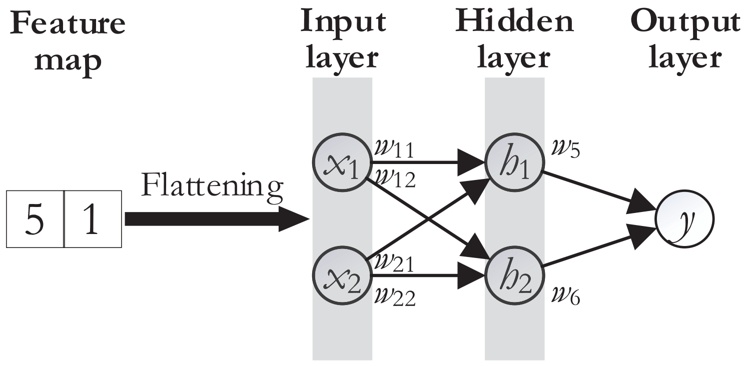

Figure 4.
An example of a fully connected process.

Variable	Equation	Calculation process
h_1	$h_1 = w_{11} \times 5 + w_{21} \times 1$	5 × 0.1 + 1 × 0.3 = 0.8
h_2	$h_2 = w_{12} \times 5 + w_{22} \times 1$	5 × 0.2 + 1 × 0.4 = 1.4
y	$\mathrm{y} = w_5 \times h_1 + w_6 \times h_2$	0.8 × 0.5 + 1.4 × 0.6 = 1.24
Loss	$\mathrm{Loss} = \frac{1}{2} \times \left(y_{\mathrm{actual}} - y_{output}\right)^2$	1/2 × (0.24–1.24)^2 = 0.5
w_5’	$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial y} \times \frac{\partial y}{\partial w_5} = 2 \times \frac{1}{2} \times \left(y_{\mathrm{actual}} - y_{output}\right) \times h_1 \times (-1)$	2 × 1/2 × (0.24–1.24) × 0.8 × (−1) = 0.8
	$w_5’ = w_5 - \eta \frac{\partial L}{\partial w5}$	0.5–0.1 × 0.8 = 0.42
w_6’	$\frac{\partial L}{\partial w_6} = \frac{\partial L}{\partial y} \times \frac{\partial y}{\partial w_6} = 2 \times \frac{1}{2} \times \left(y_{\mathrm{actual}} - y_{output}\right) \times h_2 \times (-1)$	2 × 1/2 × (0.24–1.24) × 1.8 × (−1) = 1.8
	$w_6' = w_6 - \eta \frac{\partial L}{\partial w6}$	0.6–0.1 × 1.8 = 0.42
w_{11}’	$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial y} \times \frac{\partial y}{\partial \mathrm{h}_1} \times \frac{\partial h_1}{\partial w_{11}} = 2 \times \frac{1}{2} \times \left(y_{\mathrm{actual}} - y_{output}\right) \times (-1) \times \mathrm{w}_5 \times x_1$	2 × 1/2 × (0.24–1.24) × (−1) × 0.5 × 5 = 2.5
	$w_{11}' = w_{11} - \eta \frac{\partial L}{\partial w11}$	0.1–0.1 × 2.5 = −0.15
w_{12}’	$\frac{\partial L}{\partial w_{12}} = \frac{\partial L}{\partial y} \times \frac{\partial y}{\partial \mathrm{h}_2} \times \frac{\partial h_2}{\partial w_{12}} = 2 \times \frac{1}{2} \times \left(y_{\mathrm{actual}} - y_{output}\right) \times (-1) \times \mathrm{w}_6 \times x_1$	2 × 1/2 × (0.24–1.24) × (−1) × 0.6 × 5 = 3
	$w_{12}' = w_{12} - \eta \frac{\partial L}{\partial w12}$	0.2–0.1 × 3 = −0.1
w_{21}’	$\frac{\partial L}{\partial w_{21}} = \frac{\partial L}{\partial y} \times \frac{\partial y}{\partial \mathrm{h}_1} \times \frac{\partial h_1}{\partial w_{21}} = 2 \times \frac{1}{2} \times \left(y_{\mathrm{actual}} - y_{output}\right) \times (-1) \times \mathrm{w}_5 \times x_2$	2 × 1/2 × (0.24–1.24) × (−1) × 0.5 × 1 = 0.5
	$w_{21}' = w_{21} - \eta \frac{\partial L}{\partial w21}$	0.3–0.1 × 0.5 = 0.25
w_{22}’	$\frac{\partial L}{\partial w_{22}} = \frac{\partial L}{\partial y} \times \frac{\partial y}{\partial \mathrm{h}_2} \times \frac{\partial h_2}{\partial w_{22}} = 2 \times \frac{1}{2} \times \left(y_{\mathrm{actual}} - y_{output}\right) \times (-1) \times \mathrm{w}_6 \times x_2$	2 × 1/2 × (0.24–1.24) × (−1) × 0.6 × 1 = 0.6
	$w_{22}' = w_{22} - \eta \frac{\partial L}{\partial w22}$	0.4–0.1 × 0.6 = 0.34
Updated feature map	$h_1' = y_{output} \times w_5'$	1.24 × 0.42 = 0.5208
	$h_2' = y_{output} \times w_6'$	1.24 × 0.42 = 0.5208
	$x_1' = h_1' \times w_{11}' + h_2' \times w_{12}'$	0.5208 × (−0.15) + 0.5208 × (−0.1) = 0.1302
	$x_2' = h_1' \times w_{21}' + h_2' \times w_{22}'$	0.5208 × 0.5+ 0.5208 × 0.6 = 0.57288

Table 2.
Detailed calculation process of feature map updating using BP.

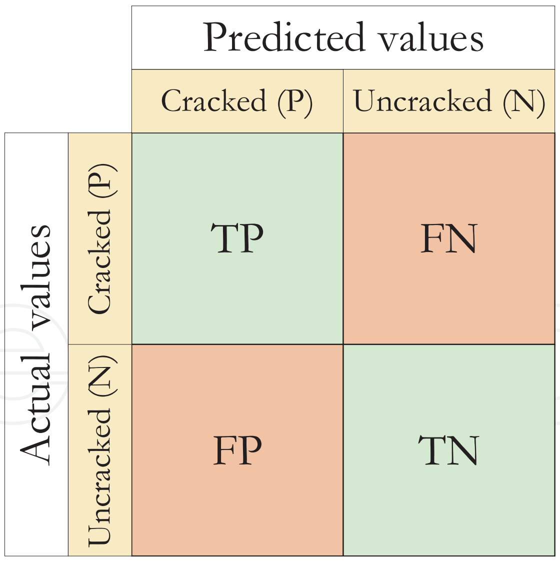

Figure 5.
An example of a fully connected process.

performance of precision and recall. A CNN with a high F1-score indicates stronger robustness.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \tag{6}$$

$$Precision = \frac{TP}{TP + FP} \times 100 \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \times 100 \tag{8}$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \times 100 \tag{9}$$

For example, the prepared dataset contains 10,000 photos, with 32,000 and 7000 cracked surface images and uncracked surface images, respectively. After CNN processing, 2700 images are correctly predicted as cracked surfaces, 300 images out of the 3000 real cracked surfaces are wrongly predicted as uncracked surfaces. 6500 images are correctly predicted as uncracked surfaces, and 500 images out of the 7000 uncracked surfaces are wrongly predicted as cracked surfaces. Then, based on above-mentioned concepts, the values of TP, FN, FP, TN is 2700, 300, 500, 6500, respectively. **Table 3** shows the details of the accuracy, precision, recall, and F1 score calculations.

Variable	Equation	Calculation process
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN} \times 100$	(2700 + 6500) /(2700 + 300 + 500 + 6500) × 100 = 92%
Precision	$\frac{TP}{TP+FP} \times 100$	2700/(2700 + 500) × 100 = 84.375%
Recall	$\frac{TP}{TP+FN} \times 100$	2700/(2700 + 300) × 100 = 90%
F1-score	$2 \times \frac{Precision \times Recall}{Precision+Recall} \times 100$	2 × ((0.84375 × 0.9)/(0.84375 + 0.9)) × 100 = 87.23%

Table 3.
Detailed calculation process of accuracy, precision, recall, and F1 score.

3. Example of concrete crack inspection using CNN

3.1 Textbook example of crack inspection using CNN

This chapter provides an example of how convolution, pooling, fully connected, and benchmarking can be demonstrated in real-world concrete crack inspection using CNN. The above-mentioned calculation was carried out using the Python programming language and the Pytorch package.

3.1.1 Dataset

In this example, the input images were gathered from Kaggle, the world's most well-known data science community. Kaggle allows access to thousands of public datasets covering a wide range of topics, including medical, agriculture, and construction [9]. By searching "concrete crack" in Kaggle datasets module, 12 datasets were found. The "SDNET2018" dataset was chosen from among them since it comprises sufficient and clean concrete surface images with and without cracks [10]. In "SDNET2018", 56,096 images were captured in the Utah State University Campus using a 16-megapixel Nikon digital camera, including 54 bridge decks, 72 walls, and 104 pavements. In this example, only images of walls and pavements were used to demonstrate the comparison analysis between manual inspection and CNN-based automatic inspection. Therefore, 42,472 images were used as training and testing dataset. Among them, 6459 cracked concrete surfaces are considered as positive class. The captured cracks are as narrow as 0.06 mm and as wide as 25 mm, while 36,013 uncracked concrete surfaces are considered as negative class. Images in this dataset contain a range of impediments, such as shadows, surface roughness, scaling, edges, and holes. The diverse photographing backgrounds contribute to ensuring the robustness of the designed CNN architecture. At a ratio of 80/20, the cracked and uncracked concrete photos were randomly separated into training and testing datasets. The input images' pixels were standardized to $227 \times 227 \times 3$ for AlexNet, and $224 \times 224 \times 3$ for VGG16. **Table 4** shows the details of the input images. **Figure 6** shows the examples of the input images.

3.1.2 CNN architecture

In this section, two pre-trained CNN networks, AlexNet and VGG16, were introduced to illustrate CNN computation process. AlexNet was designed as an eight-layer architecture. VGG16 has a depth that is two twice that of AlexNet. According to [11, 12], the depth of CNN network has a significant impact on model performance.

	Total dataset	Training dataset	Testing dataset
Total images	42,472	33,978	8494
Cracked images	6227	4986	1238
Non-cracked images	36,245	28,992	7256
Image pixels		AlexNet: $227 \times 227 \times 3$ VGG16: $224 \times 224 \times 3$	

Table 4.
Details of prepared dataset.

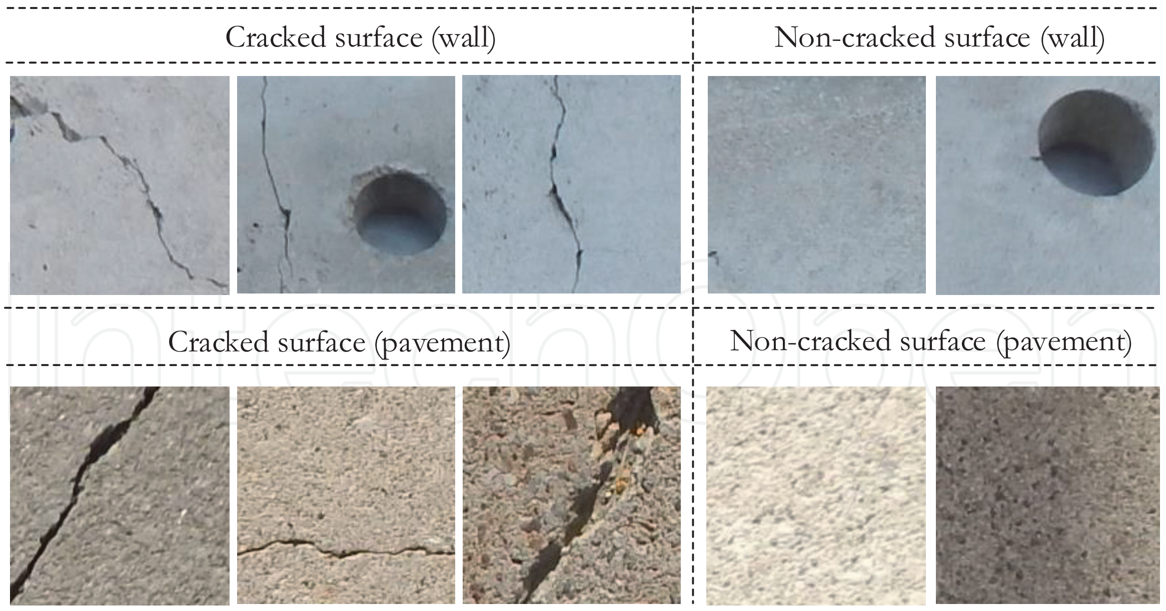
Cracked surface (wall)
Non-cracked surface (wall)
Cracked surface (pavement)
Non-cracked surface (pavement)

Figure 6.
Examples of cracked and non-cracked surface.

Therefore, by training and testing the prepared dataset with AlexNet and VGG16, the comparison of network depth to prediction performance and computation cost can be further highlighted.

1. AlexNet architecture

The AlexNet architecture, developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton in 2012, is considered one of the most influential CNN architectures [13]. AlexNet consists of five convolution layers and three fully-connected layers. The max-pooling layers follow the first, second, and fifth convolution layers. AlexNet was designed to predict 1000 object classifications. 1.2 million images with a pixel size of 2,242,243 were used as input images. As a result, 60 million parameters and 650,000 neurons are included in the computation process. The details of the AlexNet architecture are shown in **Figure 7**.

In the first convolution stage, 96 convolution filters with size of 11×11 were applied; they move with a stride with four pixels. The size of pooling filters is 3×3. The pooling filters move with a stride of two. It is worth noticing that the error rate can be reduced by applying overlapping pooling technique (the size of pooling filters is smaller than its stride). In the second convolution stage, the size of convolution filters becomes smaller from 11×11 to 5×5 while its number becomes larger from 96 to 256. The convolution filters in the third and the fourth convolution stage keep minimizing, from 5×5 to 3×3, while its number keeps increasing from 256 to 384. In the last convolution stage, the size of convolution filters remains same as 3×3, and its number turns back to 256. The size and stride of pooling filters also remain the same in the second and fifth convolution stage. Finally, 4096 neurons are included for both first and second fully-connected layers. The final fully-connected layer contains 1000 neurons to output the probabilities of 1000 classifications. The 1000 neurons are activated by softmax function.

The outputs of each convolution and fully-connected layer are activated by a non-linear function, namely the Rectified Linear Units (ReLU) [14]. It is proved in

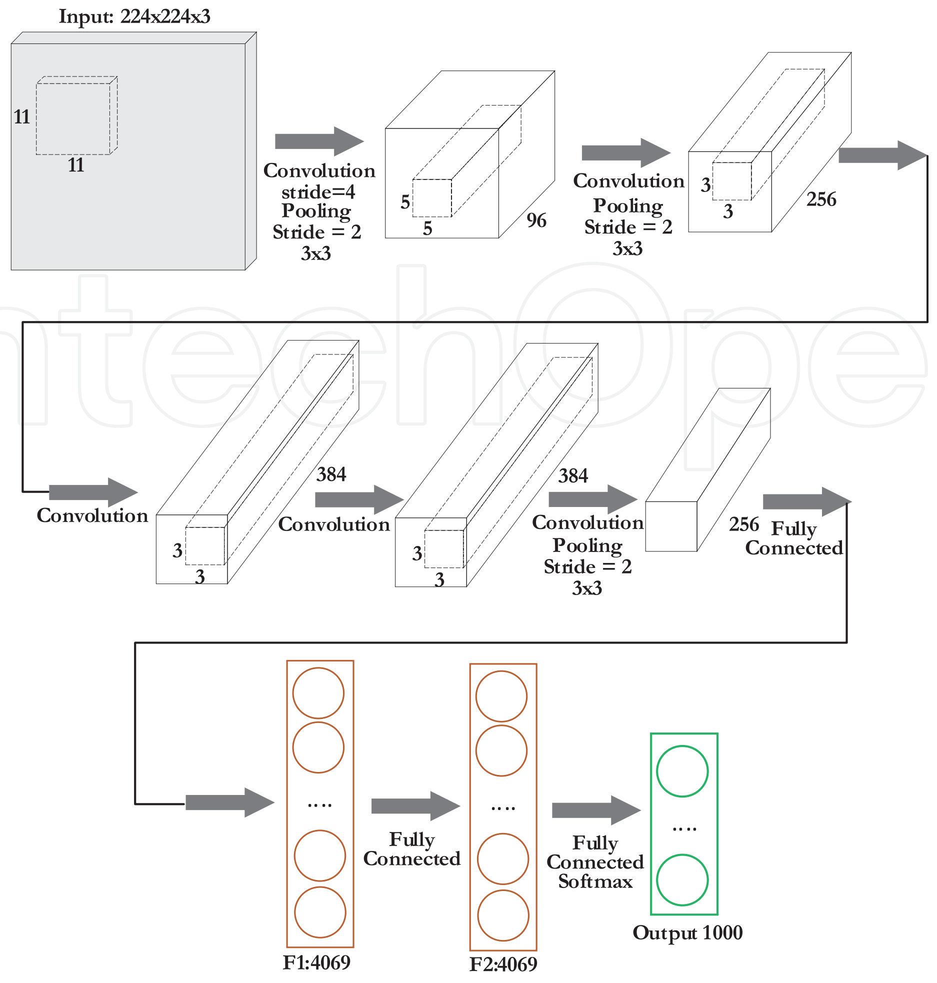

Figure 7.
Details of AlexNet architecture.

AlexNet that using ReLU instead of other activation functions effectively solves the overfitting problem and improves computation efficiency. Especially for the larger architectures trained on larger datasets. The local response normalization [15] technique (LRN) is also applied following ReLUs to reduce the error rate. Moreover, to avoid overfitting, drop-out techniques [16] are also applied in the first two fully-connected layers. The dropout criteria was set at 0.5.

AlexNet was computed using SGD. The batch size, momentum [17], and weight decay [18] were set as 128, 0.9, and 0.0005, respectively. The learning rate was set as 0.00001. AlexNet was computed for roughly 90 periods in NVIDIA GTX 580 3GB GPUs. As a result, the error rate of AlexNet on test set of top-1 and top-5 achieved 37.5% and 17.0%, which was 10% lower than the out-performed CNN architecture at that time.

2. VGG16 architecture

VGG16, designed by Karen Simonyan and Andrew Zisserman in 2015, was developed to investigate the influence of convolution network depth on prediction accuracy in larger datasets [19]. Therefore, VGG16 was designed as a deep architecture with 16 weight layers, including 13 convolution layers and three fully-connected layers. Convolution layers in VGG16 are presented as five convolution blocks. The details of the VGG16 architecture are shown in **Figure 8**.

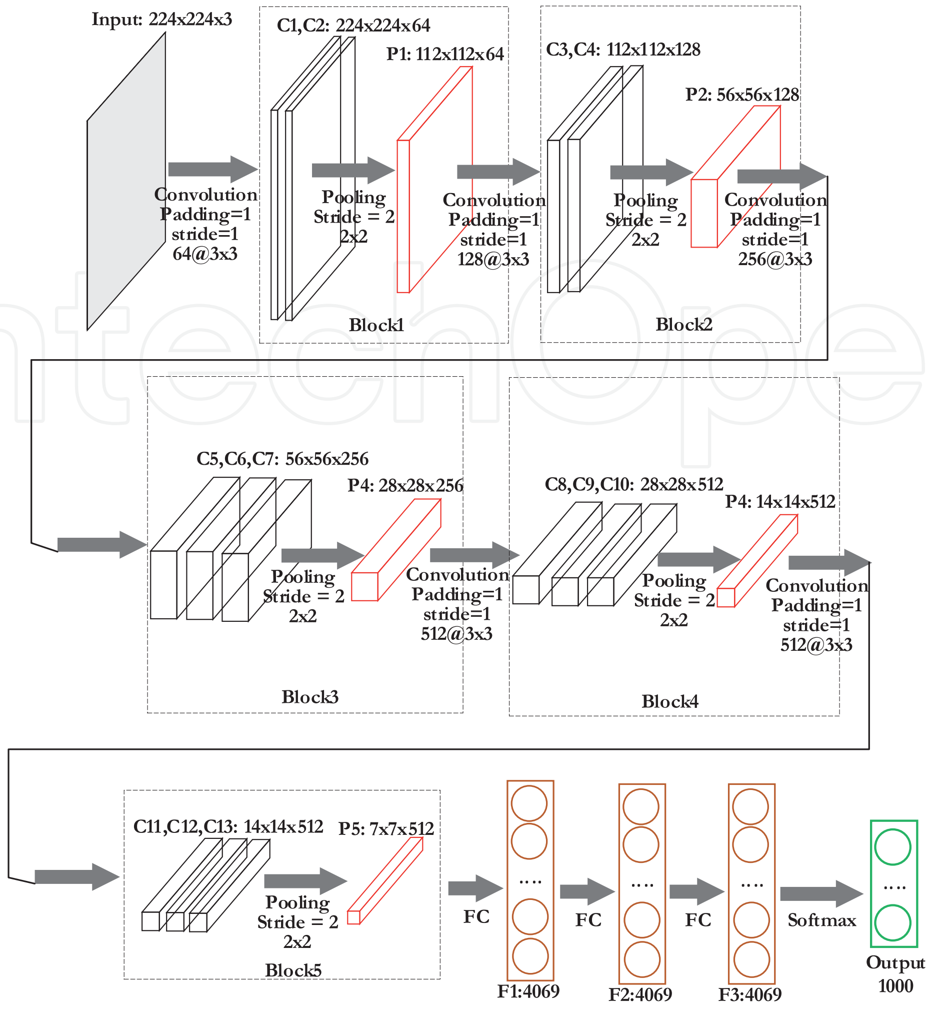

Figure 8.
Details of VGG16 architecture.

As seen from **Figure 8**, there are two convolution layers in the first two convolution blocks, respectively, and three convolution layers in the following three convolution blocks, respectively. The size of all convolution filters is uniformly 3 × 3. All the convolution filters move with a stride of one. The number of convolution filters increases gradually from 64 to 128, 256, and 512 in the five convolution blocks. To preserve information about image boundaries as completely as possible, spatial padding is applied [20]. As with AlexNet, ReLU is applied as a non-linearity function for convolution and fully-connected outputs to avoid overfitting problems. However, unlike in AlexNet, LRN is not used in VGG16 because the authors stated that LRN has no influence on model performance and increases memory consumption and computation time.

Five max-pooling layers follow the last convolution layer in each block. The max-pooling filters are uniformed with a size of 2 × 2, and a stride of two. As with AlexNet, the first two fully-connected layers have 4096 neurons and 1000 output neurons. The output neurons are activated by softmax. To avoid overfitting problems, drop-out technique is also applied in the first two fully-connected layers. The dropout ratio is set at 0.5. It can be concluded that the most important novelty of VGG16 compared with AlexNet are: (1) the designed deep architecture; (2) the uniformed and small size convolution filters.

In the training process, the training batch size, momentum, weight decay, and learning rate were set as 256, 0.9, and 0.0005, 0.0001, respectively. As a result, the top-1 and top-5 errors of VGG16 achieved 24.4% and 7.2%, which is 13% and 9.8% lower than AlexNet. The result proved that the deep architecture and small convolution filters have positive influences on CNN performance.

3.1.3 Training and benchmarking

Finally, the prepared dataset mentioned in Section 3.3.1 was used to train and test AlexNet and VGG16, respectively. The training and testing process was conducted in Kaggle kernels [21]. Kaggle kernel, provided by Kaggle community, is a virtual environment equipped with NVIDIA Tesla K80, a dual GPU design, and 24GB of GDDR5 memory. This high computing performance enables 5–10 times faster training and testing processes than CPU-only devices. Both AlexNet and VGG16 were trained using SGD. Batch size was and learning rate set as 64, 0.0001, respectively. To avoid overfitting problem, dropout was applied at the fully-connected stage, dropout probability was set as 0.5.

Python was used to program the computing process. Pytorch library was imported. The whole computation time of AlexNet was roughly 2 h, and 4 h for VGG16. The model's performance in the training and testing datasets is shown in **Figures 9** and **10**,

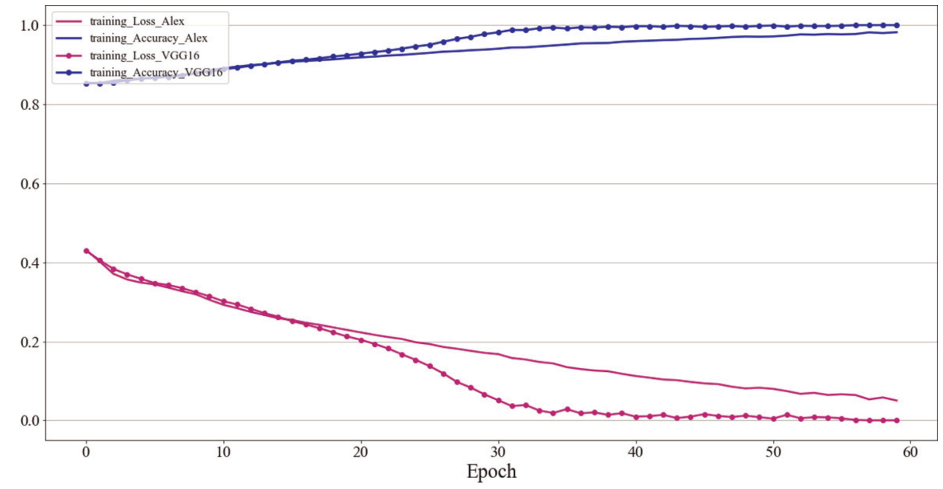

Figure 9.
Training loss and accuracy of AlexNet and VGG16.

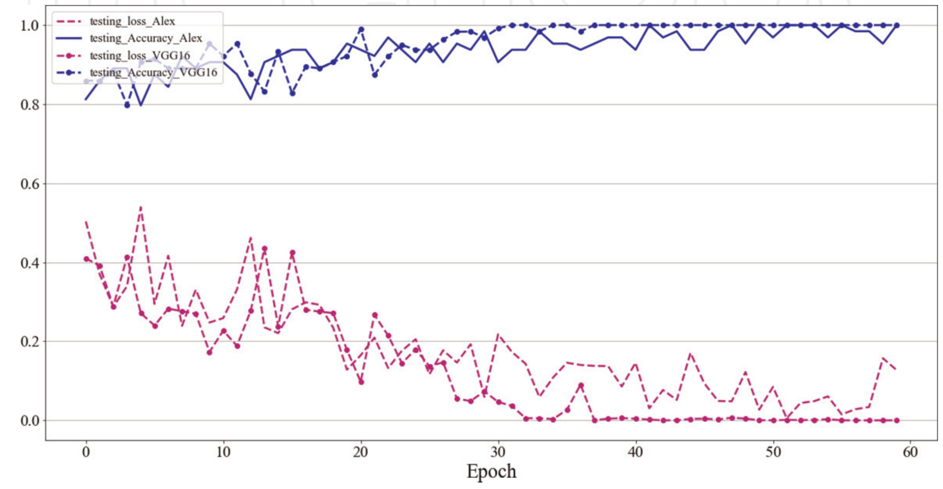

Figure 10.
Testing loss and accuracy of AlexNet and VGG16.

respectively. The training and testing loss and accuracy values are represented on the vertical axis, while the processing epochs are represented on the horizontal axis. Since the loss and accuracy variation remained consistent after the 60th epoch overtraining the model could lead to an overfitting problem [22]. The training epoch was set to 60 epochs.

As shown in **Figure 9**, both AlexNet and VGG16 converged successfully. The training loss for AlexNet reduced steadily from 0.43 to 0.05 in the 58th epoch and then remained constant in subsequent epochs. Similarly, at the 58th epoch, AlexNet's training accuracy increased from 0.85 to 0.98. At the 35th epoch, the training loss for VGG16 dropped from 0.42 to 0.01 and subsequently stayed steady at approximately 0.008–0.01 in following epochs. At the 34th epoch, the training accuracy of VGG16 increased from 0.85 to 0.99 and then remained at 0.99. The results revealed that VGG16 performed better during the training procedure. VGG16's convergence speed is roughly two times that of AlexNet. VGG16's minimum training loss is 0.04 lower than AlexNet's, while its maximum accuracy is 0.01 times higher. It is observed that deeper CNN designs assist in the faster processing of larger datasets, which contributes to producing more trustworthy weights and biased matrices. These results are in accordance with those proposed by [23].

Figure 10 shows the loss and accuracy variations of AlexNet and VGG16 in the testing dataset. The testing loss and accuracy consist of the fluctuation tendency of training loss and accuracy. It indicated that neither AlexNet nor VGG16 had overfitting or underfitting problems. VGG16 also out-performed AlexNet in the testing process. AlexNet and VGG16 have minimum testing losses of 0.01 and 0.00003, respectively. AlexNet's maximum accuracy was 0.98, and VGG16's was 0.99. In the testing dataset, VGG16 converges at the 34th epoch, which is nearly 2 times faster than AlexNet.

The confusion matrix of AlexNet and VGG16 is shown in **Table 5**. It can be shown that the accuracy scores of AlexNet and VGG16 are nearly identical, indicating that AlexNet and VGG16 have similar prediction abilities for cracked and uncracked concrete surfaces. VGG16 has a precision and recall of 96.5% and 89.6%, respectively, which is nearly 1% and 5% greater than AlexNet. The results show that VGG16 outperforms AlexNet for predicted positive variables (cracked surfaces). Meanwhile, more cracked images from actual datasets can be correctly identified by applying VGG16. AlexNet and VGG16 have F1-scores of 89.6% and 92.9%, respectively, indicating that the VGG16 model is more robust.

	AlexNet	**VGG16**
TP	5242	5579
FN	985	648
TN	36,007	36,040
FP	238	205
Accuracy	0.971204558	0.97991618
Precision	0.956569343	0.9645574
Recall	0.84181789	0.895937048
F1-score	0.895532587	0.928981767

Table 5.
Confusion matrix of AlexNet and VGG16.

In conclusion, VGG16 demonstrates better performance. Since it is important to avoid ignoring any cracked surfaces, the model with the highest recall and F1-score is more worthwhile. Meanwhile, AlexNet is also a preferable option when the number of cracked and uncracked images is balanced because it shows a similar accuracy score as VGG16 and has a lower computation cost.

3.2 Comparison of CNN and manual inspection

During on-site construction quality management process, quality control managers (QCM) or registered inspectors (RI) are responsible for personally inspecting and reporting quality problems with forms, reports, and photocopies. According to the Mandatory Building Inspection Scheme (MBIS) and related contract regulations, QCMs and RIs are obliged to examine cracks and other defects in building components visually or with non-destructive equipment [24]. For example (1) cracks on the structural components, e.g., structural beam, column, (2) cracks on the external finishes, e.g., tiling, rendering, and cladding, (3) cracks on the fins, grilles, windows, curtain walls.

When using computer-vision-based inspection techniques, differently, there is no necessity for QCMs and RIs to conduct the aforementioned inspection tasks on-site. Instead, their primary responsibilities may switch to (1) taking photos or videos of building components, and (2) inputting the images and videos into pre-trained CNN models. To highlight the differences between manual and computer-vision-based crack inspection, an experiment was set up to calculate and compare inspection time and cost.

The layout of the experiment is shown in **Figure 11**. Suppose this experiment case is a 15 m $\times$ 15 m $\times$ 2 m residential building that is located in San Bernardino. The inspection items include cracks on slab, internal walls, and external walls. According to Dohm, John Carl [25], the total manual inspection time for 1600–2600ft^2 home in San Bernardino is around 13.65 h, including inspection items of building slab, shear walls, etc. The manual inspection service cost is around $85.9 per hour.

Referring to the computer-vision-based inspection process described above, the total inspection time includes the time of taking images or videos and CNN processing. Assume that the input videos are obtained with handheld camera devices while QCMs or RIs are by means of walking. Then, the time of taking videos can be considered as the time of walking.

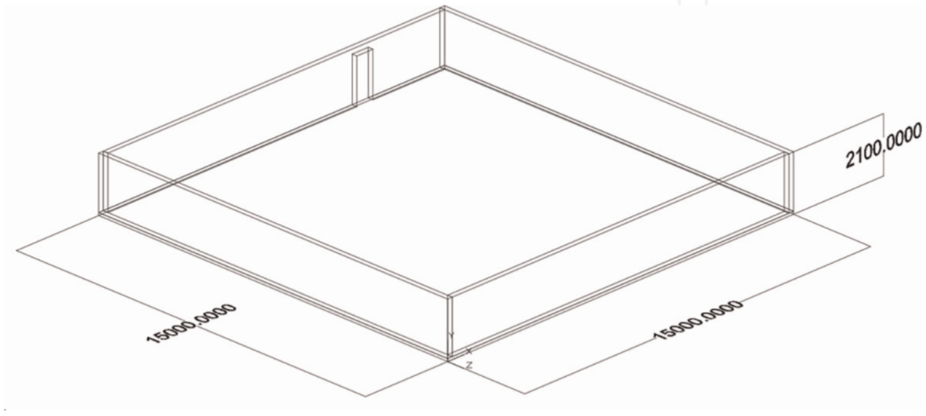

Figure 11.
Layout of the experiment case.

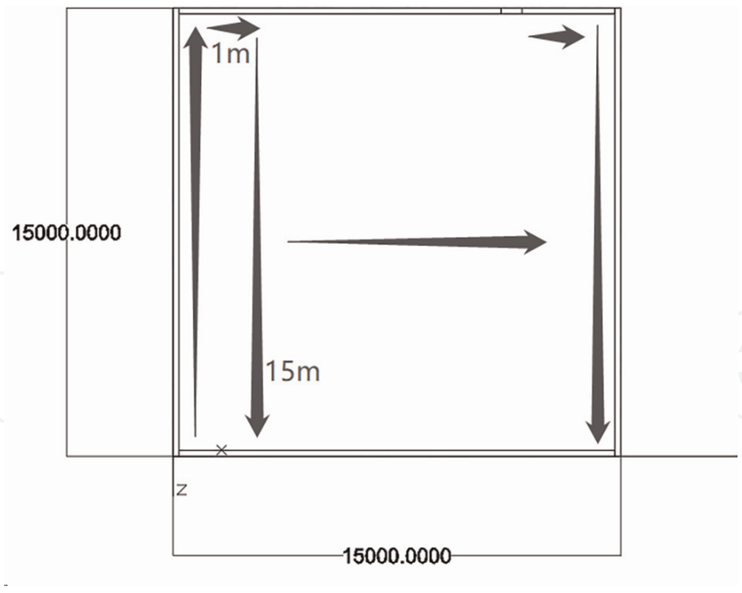

Figure 12.
Walking path of the inspectors.

Manual inspection		Computer-vision based inspection		
Time	13.65 × 3600 = 49,140 s	Time	Taking video	(1/0.1 × 15) × 15) +1/0.1 × 14 = 2390 s
			CNN processing	(2390 × 24)/100 = 573.6 s
Cost	(85.9/3600) × 49,140 = $1172.5	Cost	(573.6 + 2390) × (85.9/3600) = $70.7	

Table 6.
Time and cost of manual and computer-vision based crack inspection.

Normally, the average walking speed between the age of 20–49 is around 1.42 m/s [26]. Considering the time delays of taking videos, the walking speed can be considered as 0.1 m/s. Suppose the walking path follows an S-curve, shown in **Figure 12**.

According to [27], the universally accepted frame rate is 24 FPS per second. Suppose the inspector begins to record video while taking the first step. Then the time of captured video equals the time of walking. The number of the input images that converted from the captured video can be calculated as 2390 s × 24FPS = 57,360. According to the testing time of the textbook examples mentioned in Section 3.1, and the study outcomes of [28], the time of CNN processing is around 100 images per second. Then, the time of CNN processing can be calculated as 57,360/100 = 573.6 s. Therefore, the cost of computer-vision based crack inspection can be calculated as (2390 s + 573.6 s) × (85.9/3600) = $70.7.

Table 6 summarizes the calculation process of time and cost of manual and computer-vision-based crack inspection. It can be seen that using CNN-based technique can effectively reduce inspection time and cost. The inspection time decreases from 13.65 to 0.8 h in total, the inspection cost decreases from $1172.5 to $70.7.

4. Conclusion

To facilitate automatic building quality inspection and management, this study introduced a computer-vision-based automated concrete crack inspection technique. In order to demonstrate the computing and benchmarking process, the mathematical understanding of one of the most essential computer vision algorithms, convolution neural network, was first detailed.

The theoretical foundation was then explained using a textbook example. In this case, the input dataset "SDNET2018" was obtained from the Kaggle community. A digital camera was used to acquire the 56,096 photos from the Utah State University campus. To train the input images, the two most basic CNN architectures, AlexNet and VGG16, were chosen. The Pytorch library was used to carry out the training process in the Kaggle kernel. The model's performance was evaluated using a confusion matrix. The results revealed that the prediction accuracy of AlexNet and VGG16 is nearly identical. However, VGG16's precision and recall are higher than AlexNet's, indicating that VGG16 has a stronger capacity to identify cracked surfaces. VGG16's F1 score is also greater than AlexNet's, signifying that VGG16 is more robust. VGG16 is deemed to have a better significance since it has higher precision, recall, and F1-score, which is crucial when distinguishing cracked and uncracked surfaces. When the ratio of cracked and uncracked images is almost the same, however, AlexNet is a feasible alternative because of its high accuracy score and low computation cost. It's worth noting that, when compared to shallow CNN architectures, deeper and broader CNN architectures outperform shallow CNN architectures for larger datasets.

Next, an experimental case was designed to compare manual and computer-vision-based crack inspection in terms of time and cost. The results showed that the efficiency and cost-effectiveness can be effectively improved when adopting computer-vision-based techniques. The inspection time and cost or the designed case can nearly decrease from 13.65 to 0.8 h, and from $1172.5 to $70.7, respectively.

The findings help to demonstrate the computer-vision-based quality inspection technique in both theory and practice. Although the recently developed computer-vision-based technology improves the efficiency, cost-effectiveness, and safety of human quality inspection, it still relies primarily on the collected image quality. Some concrete surface images are difficult to capture in real-life situations, including among others high-rise buildings, component corners, and buildings in extremely harsh environments. To address this issue, robotics techniques are growing rapidly as a means of upgrading computer-vision-based quality inspection [29]. Previous research has begun to use mobile robots, such as UAVs in order to gather surface images [30–32]. Some studies have focused on exploring robotic inspection systems to raise the automatic level of quality inspection [33, 34]. Therefore, merging robotics and computer vision approaches may be considered as a worthwhile future research direction to improve the efficiency and accuracy of manual quality control and management.

Acknowledgements

The authors highly appreciate the full support funding of the full-time PhD research studentship under the auspice of the Department of Building and Real Estate, The Hong Kong Polytechnic University, Hong Kong. The authors would like to express their deepest gratitude to Prof. Heng Li for his guidance. Finally, the authors would like to acknowledge the research team members (Mr. King Chi Lo, Mr. Qi Kai) and anyone who provided help and comments to improve the content of this article.

Conflict of interest

All authors declare that they have no conflicts of interest.

List of abbreviations

PASS	Building Performance Assessment Scoring System
MBIS	Mandatory Building Inspection Scheme
MWIS	Mandatory Window Inspection Scheme
NDT	Non-Destructive
CNN	Convolutional Neural Network
RGB	Red, Green, and Blue
ANN	Artificial Neural Network
BP	Back-Propagation
MSE	Mean Square Error
SGD	Stochastic Gradient Descent
ReLU	Rectified Linear Units
LRN	Local Response Normalization
QCM	Quality Control Managers
RI	Registered Inspectors

Author details

Siwei Chang and Ming-Fung Francis Siu*
The Hong Kong Polytechnic University, Hung Hom, Hong Kong

*Address all correspondence to: francis.siu@polyu.edu.hk

References

[1] Driscoll R. Assessment of damage in low-rise buildings, with particular reference to progressive foundation movement. In: Digest. London: H.M.S. O.; 1981. p. 251

[2] Chitte CJ, Sonawane YN. Study on causes and prevention of cracks in building. International Journal for Research in Applied Sciences and Engineering Technology. 2018;**6**(3): 453-461. DOI: 10.22214/ijraset.2018.3073

[3] Kim B, Cho S. Image-based concrete crack assessment using mask and region-based convolutional neural network. Structural Control and Health Monitoring. 2019;**26**:e231. DOI: 10.1002/ stc.2381

[4] Rao A, Nguyen T, Palaniswami M, Ngo T. Vision-based automated crack detection using convolutional neural networks for condition assessment of infrastructure. Structural Health Monitoring. 2020;**20**:1475921720 96544. DOI: 10.1177/14759217209 65445

[5] Vandoni HT, Carlo E. Computer Vision: Evolution and promise. 19th CERN School of Computing. Geneva: CERN; 1996. pp. 21-25. DOI: 10.5170/ CERN-1996-008.21 ISBN 978-9290 830955

[6] Feng X, Jiang Y, Yang X, Du M, Li X. Computer vision algorithms and hardware implementations: A survey. Integration. 2019;**69**:309-320. DOI: 10.1016/j.vlsi.2019.07.005

[7] Yamashita R, Nishio M, Do RKG, Togashi K. Convolutional neural networks: an overview and application in radiology. Insights into Imaging. 2018; **9**(4):611-629. DOI: 10.1007/s13244-018-0639-9

[8] Stehman SV. Selecting and interpreting measures of thematic classification accuracy. Remote Sensing of Environment. 1997;**62**(1): 77-89. DOI: 10.1016/S0034-4257(97) 00083-7

[9] Banachewicz K, Massaron L. Data Analysis and Machine Learning with Kaggle: How to Compete on Kaggle and Build a Successful Career in Data Science. Birmingham, United Kingdom: Publisher Packt Publishing Limited. 2021. Available from: https://www. bookdepository.com/Data-Analysis-Machine-Learning-with-Kaggle-Konrad-Banachewicz/9781801817479

[10] Dorafshan S, Robert JT, Marc M. SDNET2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks. Data in Brief. 2018;**21**: 1664-1668. DOI: 10.1016/j.dib.2018. 11.015

[11] OrShea A, Lightbody G, Boylan G, Temko A. Investigating the impact of CNN depth on neonatal seizure detection performance. In: 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC); 18–21 July 2018; USA. New York: IEEE; 2018. pp. 5862-5865

[12] Pasupa K, Sunhem W. A comparison between shallow and deep architecture classifiers on small dataset. In: 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE); 5–6 October 2016; Indonesia. New York: IEEE; 2016. pp. 1-6

[13] Krizhevsky A, Ilya S, Hinton GE. Imagenet classification with deep convolutional neural networks.

Advances in Neural Information Processing Systems. 2018;**25**:1097-1105. DOI: 10.1145/3065386

[14] Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. In: Proceedings 27th International conference on machine learning; 21–24 June. Israel: International Machine Learning Society; 2010. pp. 417-425

[15] Kim GB, Jung KH, Lee Y, Kim HJ, Kim N, Jun S, et al. Comparison of shallow and deep learning methods on classifying the regional pattern of diffuse lung disease. Journal of Digital Imaging. 2018;**31**(4):415-424. DOI: 10.1007/s10278-017-0028-9

[16] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research. 2014;**15**(1):1929-1958. DOI: 10.5555/2627435.2670313

[17] Smith LN A Disciplined Approach to Neural Network Hyper-Parameters: Part 1: Learning Rate, Batch Size, Momentum, and Weight Decay. arXiv preprint arXiv:1803.09820. 2018

[18] Gnecco G, Sanguineti M. The weight-decay technique in learning from data: An optimization point of view. Computational Management Science. 2008;**6**(1):53-79. DOI: 10.1007/s10287-008-0072-5

[19] Simonyan K & Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv; 1409.1556. 2014

[20] He K, Zhang X, Ren S, Sun J. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2015;**37**(9): 1904-1916. DOI: 10.1109/TPAMI.2015.2389824

[21] Banachewicz K. Data Analysis and Machine Learning with Kaggle. Birmingham: Packet Publishing Limited; 2021

[22] Al Haque AF, Rahman MR, Al Marouf A, Khan MAAA. Computer vision system for Bangladeshi local mango breed detection using convolutional neural network (CNN) models. In: 4th International Conference on Electrical Information and Communication Technology (EICT); 20–22 December 2019; Bangladesh. New York: IEEE; 2019. pp. 1-6

[23] Zheng Z, Yang Y, Niu X, Dai H, Zhou Y. Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids. IEEE Transactions on Industrial Informatics. 2018;**14**(4):1606-1615. DOI: 10.1109/tii.2017.2785963c

[24] Buildings Department. Code of Practice for Mandatory Building Inspection Scheme and Mandatory Window Inspection Scheme. Hong Kong: Hong Kong Government. 2012. Available from: https://www.bd.gov.hk/doc/en/resources/codes-and-references/code-and-design-manuals/CoP_MBIS_MWISe.pdf, https://www.bd.gov.hk/en/safety-inspection/mbis/index.html

[25] Dohm JC. Building Inspection Fee Analysis. Theses Digitization Project. San Bernardino, California, United States: California State University. 2007. Available from: https://scholarworks.lib.csusb.edu/etd-project/3249

[26] Mohler BJ, Thompson WB, Creem-Regehr SH, Pick HL, Warren WH. Visual flow influences gait transition speed and

preferred walking speed. Experimental Brain Research. 2008;**181**(2):221-228. DOI: 10.1007/s00221-007-0917-0

[27] Apple Inc. Final Cut Pro User Guide. Apple, One Apple Park Way, Cupertino, CA 95014, United States: Apple Inc. 2021. Available from: https://support.apple.com/en-hk/guide/final-cut-pro/ver917522c9/mac

[28] Ma D, Fang H, Wang N, Xue B, Dong J, Wang F. A real-time crack detection algorithm for pavement based on CNN with multiple feature layers. Road Materials and Pavement Design. 2021:1-17. DOI: 10.1080/14680629.2021.1925578

[29] Chang S, Siu MFF, Li H, Luo X. Evolution pathways of robotic technologies and applications in construction. Advanced Engineering Informatics. 2022;**51**:101529

[30] Seo J, Duque L, Wacker J. Drone-enabled bridge inspection methodology and application. Automation in Construction. 2018;**94**:112-126. DOI: 10.1016/j.autcon.2018.06.006

[31] Humpe A. Bridge inspection with an off-the-shelf 360° camera drone. Drones. 2020;**4**(4):67. DOI: 10.3390/drones4040067

[32] Liu Y, Nie X, Fan J, Liu X. Image-based crack assessment of bridge piers using unmanned aerial vehicles and three-dimensional scene reconstruction. Computer-aided Civil and Infrastructure Engineering. 2020;**35**(5):511-529. DOI: 10.1111/mice.12501

[33] La H, Gucunski N, Dana K, Kee S. Development of an autonomous bridge deck inspection robotic system. Journal of Field Robotics. 2017;**34**(8):1489-1504. DOI: 10.1002/rob.21725

[34] Montero R, Menendez E, Victores JG, Balaguer C. Intelligent robotic system for autonomous crack detection and characterization in concrete tunnels. In: 2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC); 26–28 April 2017; Portugal. New York: IEEE; 2017. pp. 316-321