sustainability

Article

Crack Detection in Concrete Structures Using Deep Learning

Vaughn Peter Golding, Zahra Gharineiat, Hafiz Suliman Munawar * and Fahim Ullah

School of Surveying and Built Environment, University of Southern Queensland, Springfield Central, QLD 4300, Australia; vaughn.golding@gmail.com (V.P.G.); zahra.gharineiat@usq.edu.au (Z.G.); fahim.ullah@usq.edu.au (F.U.)
* Correspondence: hafizsuliman.munawar@usq.edu.au

Abstract: Infrastructure, such as buildings, bridges, pavement, etc., needs to be examined periodically to maintain its reliability and structural health. Visual signs of cracks and depressions indicate stress and wear and tear over time, leading to failure/collapse if these cracks are located at critical locations, such as in load-bearing joints. Manual inspection is carried out by experienced inspectors who require long inspection times and rely on their empirical and subjective knowledge. This lengthy process results in delays that further compromise the infrastructure's structural integrity. To address this limitation, this study proposes a deep learning (DL)-based autonomous crack detection method using the convolutional neural network (CNN) technique. To improve the CNN classification performance for enhanced pixel segmentation, 40,000 RGB images were processed before training a pretrained VGG16 architecture to create different CNN models. The chosen methods (grayscale, thresholding, and edge detection) have been used in image processing (IP) for crack detection, but not in DL. The study found that the grayscale models (F1 score for 10 epochs: 99.331%, 20 epochs: 99.549%) had a similar performance to the RGB models (F1 score for 10 epochs: 99.432%, 20 epochs: 99.533%), with the performance increasing at a greater rate with more training (grayscale: +2 TP, +11 TN images; RGB: +2 TP, +4 TN images). The thresholding and edge-detection models had reduced performance compared to the RGB models (20-epoch F1 score to RGB: thresholding −0.723%, edge detection −0.402%). This suggests that DL crack detection does not rely on colour. Hence, the model has implications for the automated crack detection of concrete infrastructures and the enhanced reliability of the gathered information.

Keywords: crack detection; convolutional neural network; image processing; deep learning; damage detection

check for updates

Citation: Golding, V.P.; Gharineiat, Z.; Munawar, H.S.; Ullah, F. Crack Detection in Concrete Structures Using Deep Learning. *Sustainability* **2022**, *14*, 8117. https://doi.org/10.3390/su14138117

Academic Editors: JeongWook Son, Byungjoo Choi and Sungjoo Hwang

Received: 1 June 2022
Accepted: 30 June 2022
Published: 2 July 2022

1. Introduction

Cracks in concrete structures are a common phenomenon associated with corrosion, chemical deterioration, and the application of adverse loading. The appearance of these cracks is a sign of stress, weakness, and wear and tear within the structure, leading to possible failure/collapse [1,2]. The significance of a crack depends on its length, width, depth, and location. Therefore, monitoring the structural health, reliability, and performance is essential for the long-term serviceability of the infrastructure. Visual inspection is the prevalent form of crack detection, which is a slow, labour-intensive task. The results of these inspections rely on the skill, experience, and subjectivity of the inspector [3,4]. Further, such methods are time-consuming and often result in late assessments of cracks near the unrepairable zone. To address this, modern methods aim to speed up the process by replacing manual processes with more sophisticated, automated crack detection. Accordingly, DL methods and computer vision approaches are being widely applied to automate crack detection. One of the key approaches used for such automation is image processing (IP) or aerial imagery. Automated crack detection using imagery reduces costs and assessment time and increases the safety and objectivity of concrete inspections [5,6].

The cracking phenomenon in concrete is complex, and it depends on a number of factors such as rate of drying, amount of drying, tensile strength and strain, elasticity, drying shrinkage, and degree of restraint. Concrete contains more water than needed for hydration; therefore, shrinkage begins when concrete starts to dry. No cracks are developed when concrete is unrestrained, but it is not possible to support structures without restraints. Different types of crack are observed in concrete structures, such as plastic-shrinkage cracking, map cracking, hairline cracking, pop-outs, scaling, spalling, D-cracking, offset cracking, and diagonal corner cracking. The structural stability and durability are not affected by most types of cracking; however, in extreme cases, it may be affected.

Researchers have explored automated crack detection in concrete infrastructures, emphasizing crack identification, categorization, crack length, and width measurement. The most used methods of crack detection are image-based (thresholding, filtering, morphological, skeletonization, etc.) [5] and machine learning (ML) (convolutional neural network (CNN), fully convolutional network (FCN), random forest, etc.) [7,8]. However, not much work is being carried out to investigate the impact of low-quality images on crack detection or how the preprocessing of images can facilitate overcoming these impacts. Hence, this research was carried out to investigate the effect of preprocessing images on pretrained CNN models.

The two main methods of automatic crack detection have achieved varying degrees of success. The DL methods of ML have demonstrated classification results above 90%, while IP methods have achieved less than 80% [9,10]. This difference in accuracy has led to DL methods becoming more common [7]. DL segmentation is the current trend in crack detection, allowing the identification of pixels as cracks and therefore allowing for crack size measurement [7]. Dung and Anh (2019) suggested that segmentation needs to be more robust, and the impact of noisy crack-like features need to be reduced in order to increase the measurement accuracy above 90%.

Improved DL methods have focused on new techniques, such as pretrained models with smaller datasets, to achieve greater accuracy [10,11]. Many DL models, such as CNN, use larger datasets to improve their results, while some researchers have proposed other methods, such as NLP and RNN. Enhancing the relevant features and removing redundant information can increase DL classification performance [12]. Bui et al. (2016) found images converted to grayscale increased the performance of a CNN with object recognition. Similarly, Xie and Richmond (2018) found that grayscale images improved the performance of a CNN in lung disease classification. For crack detection, Shahriar and Li (2020) proposed preprocessing the data to determine the image quality and improve results by enhancing the imagery [13]. The following are the possible causes of cracks in concrete structures, such as moisture, temperature, and the permeability of concrete, as shown in Figure 1, below.

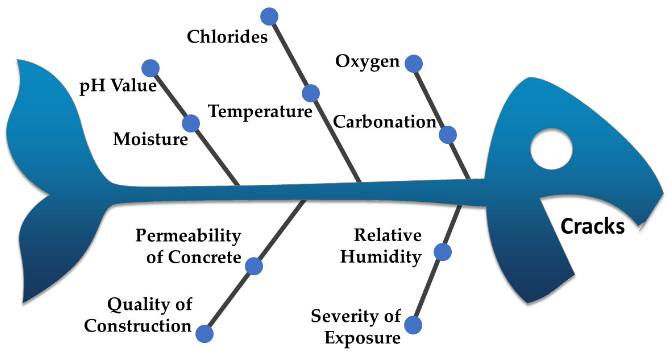


Figure 1. Possible causes of cracks in concrete structures.

Prano et al. (2022) developed an improved fracture approach to analyse the degradation of vibration characteristics for reinforced concrete beams under progressive damage [14]. The proposed framework was found to be consistent in the identification of damage at increasing damage levels that were simulated by the proposed model. The availability of mathematical models that can accurately detect damage, along with the availability of consistent data, will help to assess damage detection techniques. Similarly, De Maio et al. (2022) proposed an interelement fracture model based on the cohesive approach to simulate and investigate cracking behaviour in RC members exposed to tension and flexural loads [15]. The proposed method was able to accurately predict crack width and spacing.

IP techniques have been used to improve the performance of crack detection algorithms. The most common of these techniques are thresholding and edge detection [5]. Preprocessing images can reduce noise and complex features or highlight features used in DL models [5,14]. Therefore, the current study aims to determine if preprocessing images can improve the performance of DL for crack detection. Preprocessed RGB crack images with different IP techniques (grayscale, thresholding, and edge detection) were evaluated to estimate if the techniques could improve CNN classification performance in crack detection and gain valuable insights on the effects of processed imagery datasets on DL algorithms.

The objectives of the study are:

(i) To develop and validate a CNN suitable for crack detection;
(ii) To train the developed CNN using processed or unprocessed images, creating different models;
(iii) To analyse relative performance between the trained models in crack detection.

The rest of the study is organized as follows. In Section 2, the background literature is reviewed, and gaps are identified. The methodology for testing the preprocessing effects on DL is elaborated in Section 3. The results obtained are summarized in Section 4 and discussed in Section 5. Finally, Section 6 concludes the findings and recommends a future research direction to investigate the effects of preprocessing on DL crack detection.

2. Background Literature

IP methods routinely used preprocessing techniques to increase the accuracy. These include grayscaling, thresholding (binarization), smoothing, edge detection, and image normalization. Many of these preprocessing techniques are standard IP techniques that have been used to enhance crack identification methods [5]. The ML performance of trained models and DL methods has recently become popular. However, these methods have focused on increasing the data size to enhance performance instead of improving IP [12,15]. Hence, this study focused on different IP methods and the impact of the quality of images that can enhance the detection of cracks.

2.1. Literature Retrieval

A systematic method of literature retrieval was followed in this study. To retrieve articles related to DL/ML technologies applied to crack detection, we first consulted the primary sources: the official websites of journals and conferences. The websites chosen for the study were PubMed, MedRxiv, MDPI, Elsevier, ScienceDirect, Scopus, Google Scholar, WOS, and SSRN. The relevant articles were downloaded as PDFs and the corresponding information was entered into Excel sheets. The VOSviewer software (Version 1.6.18, Centre for Science and Technology Studies, Leiden University, Leiden, The Netherlands) tool was used to visualize the bibliometric networks from different databases. Search phrases to be used in the search engines of these websites were carefully designed to exhaust the database of each website and retrieve a maximum number of relevant articles (see Figure 2). For instance, the search phrases formulated for Category 1 (Cat-1) included phrases like “cracks,” “concrete,” and “visual inspection.” The idea was to use various sequences of keywords related to each category. Similarly, for Cat-2, the search phrases included “automated crack detection,” “image processing,” and “deep learning.” For Cat-3, we used specific keywords for each subcategory. These included “Machine learning,” “CNN,” “grayscale”, “RGB”, “Otsu method”, and “Sobel filter.” As these research articles were collected from numerous resources published in varying years, it was crucial to define some screening criteria according to which the articles were retained so that only recent and authentic articles were included in the study (Figure 3). For this purpose, we defined a timeframe for the articles, ensured the uniqueness of each article, specified authentic websites for their retrieval, and narrowed down the type of research article that was required for the study. More precisely, we defined the following assessment criteria for screening:

1. Published between 2010 and 2020;
2. English language only;
3. Article type must be a research article, review, or book chapter (letters, abstracts, and comments were not required);
4. No duplicates.

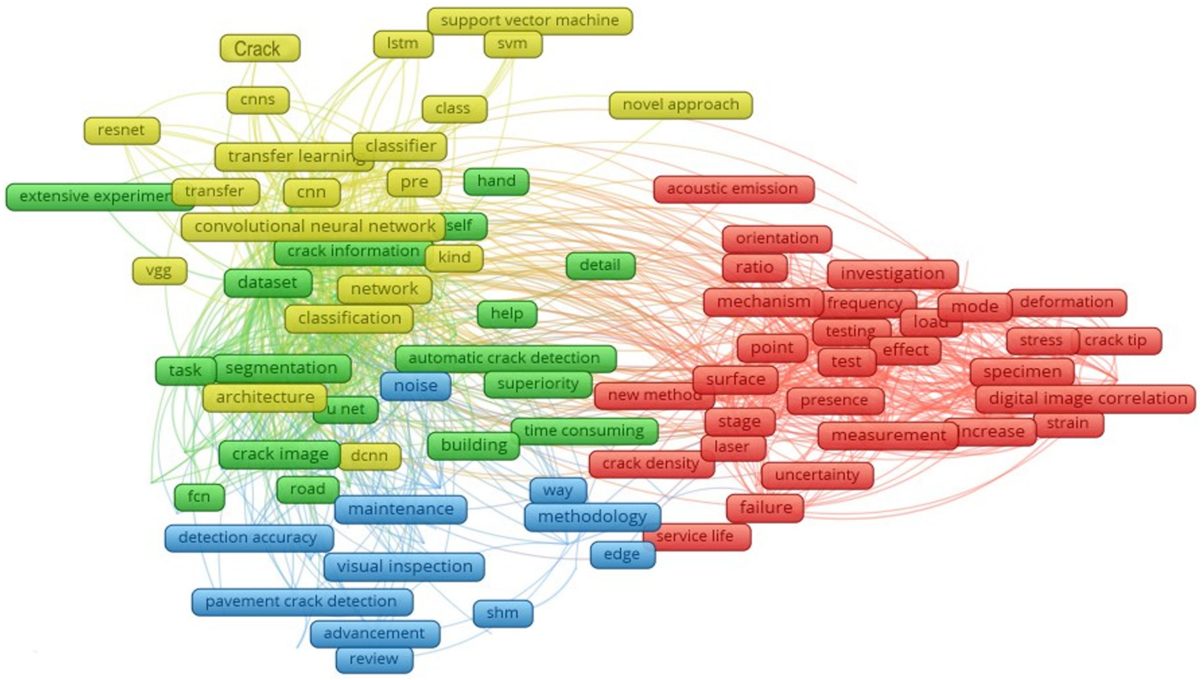


Figure 2. Keywords found in the literature related to crack detection through advanced techniques.

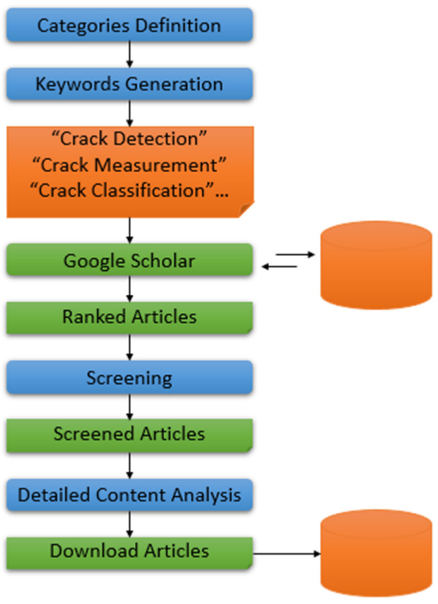


Figure 3. Article screening methodology.

2.2. *Image Processing Methods*

IP is one of the main methods of automatic crack detection. The IP methods are reliant on the pixel size of the images. Image-based crack detection commonly has four steps: (1) image acquisition, (2) preprocessing techniques, (3) IP technique (filter, morphological processing, binarization), and (4) crack quantification [1]. The architecture of this process is shown below (Figure 4).

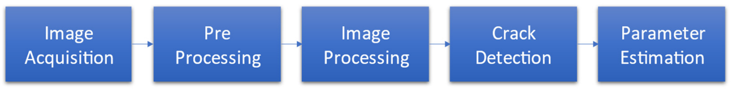


Figure 4. Image processing-based crack detection architecture.

The methods of IP are classified into different categories: integrated algorithm, morphological approach, percolation-based method, and practical technique [5,16]. The integrated algorithm method uses a preprocessing step to remove noise and enhance crack features. It uses threshold segmentation for crack identification. The morphological approach method uses mathematical morphology and curvature evaluation to detect crack structures. The percolation-based method determines if a focal pixel is part of a crack using the neighbourhood pixels to determine crack extents. While the practical technique uses the manual identification of the endpoints of the crack, in which the automatic route-finder algorithm determines the length and width of the cracks [15,16]. All the IP methods preprocess images as the first step to improve the performance of the crack detection algorithm.

The accuracy of these IP methods has been extensively studied. Most studies focus on non-invasive methods such as thresholding and feature extraction. However, it has been noted that there is very little consistency in specific IP methods [5]. Thresholding, for example, has used very simple methods, such as being fixed to 0.5 or halving the maximum image brightness; this varies in more advanced methods, such as Otsu and Niblack [5,17,18].

2.2.1. Grayscaling and Thresholding

Thresholding is the most common method used in IP for crack detection [5,19,20]. Thresholding converts a grayscale image to black and white. Grayscale images have values/brightness ranging from 0 (black) to 255 (white). Each pixel brightness is compared to a threshold value. The pixel is converted to black if the brightness is below the threshold and white if it is above the threshold (see Figure 5) [17,21]. The grayscale conversion method varies in different studies, but the common methods used are the average, luma, and luminance [22]. Thresholding is used to distinguish cracks from other background defects.

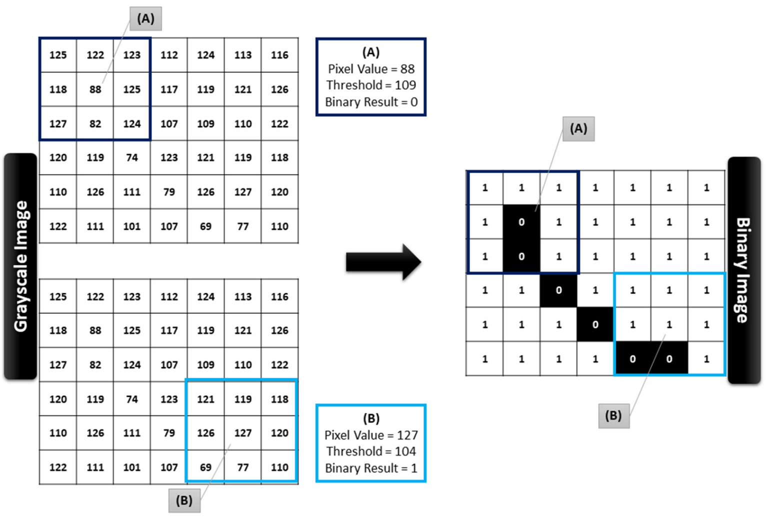

(A)
Grayscale Image
(A) Pixel Value = 88 Threshold = 109 Binary Result = 0
(B) Pixel Value = 127 Threshold = 104 Binary Result = 1
(B)
Binary Image

Figure 5. Schematic demonstration of image binarization using 3 × 3 windows.

2.2.2. Edge Detection

Edge detection methods are based on filters in an IP algorithm to enhance or detect edges. Cracks within a two-dimensional image are defined as edges or a discontinuity in the grayscale intensity field [9]. A comparison of different edge detectors, their operational method, and advantages, along with limitations, is provided in Table 1, below.

Table 1. Advantages and limitations of edge detection methods.

Edge Detector	Method	Advantages	Limitations	Reference
Roberts	Gradient-Based	- Easy and simple computation. - Edges are detected along with their orientation.	- More sensitive to noise. - Detection of edges is inaccurate. - Less reliable.	[23]
Sobel				
Prewitt				
Canny	Gaussian-Based	- Improved signal-to-noise ratio. - Suitable for noisy images, i.e., more sensitive to noisy pixels. - Accurate.	- Slow and complex. - False zero-crossing.	[16]
LoG	Gradient-Based	- The detection of edges and their orientation is simple due to the approximation of gradient magnitude is simple. - The characteristics are fixed in all directions. - Testing wide area around the pixel is possible.	- Malfunctioning at the corners, curves, and where the gray level intensity function varies. - The magnitude of edges degrades as noise increases.	[9]
DWT	Wavelet-Based	- More accurate than other methods. - Less computation.	- Application-oriented. - Complicated as compared to traditional methods.	[24]
Watershed	Gradient-Based	- Closed contours. - Less computation time. - Fast, simple, and intuitive. - Produces a complete division of the image in separated regions.	- Over segmentation. - Under segmentation.	[25]

2.3. *Traditional Machine Learning (ML) Methods*

Traditional ML methods use a predefined feature extraction stage before training the models. The most common traditional ML methods are support vector machines (SVM), artificial neural networks (ANN), random forest, clustering, Bayesian probability, and naive Bayes fusion [7,25]. Traditional ML methods have been used for both crack detection and IP with optimal parameters. These are also commonly used for predefined feature extraction [7]. However, they cannot deal with large quantities of datasets. The limitation of traditional ML methods is that they cannot learn more complex features and cannot deal with the complex information within images, such as background pavement with different lighting [7,26].

2.4. *Deep Learning-Convolutional Neural Network (CNN)*

The most common method of DL for crack detection is CNN. CNNs have been shown to outperform edge detection and ML classifiers [9] (Table 2). Pretrained CNNs have previously demonstrated an accuracy of over 98% for crack/non-crack detection. However, the CNN can only identify the image with the crack, not the crack as a pixel or feature [27,28].

Yang et al. (2019) added thermal images of heated sheet metal to a CNN to deter-mine any improvement in crack detection. It was noted that red–green–blue (RGB) could not detect steel cracks covered in pollutants or internal cracks. However, the comparison with the traditional visual method was not carried out to measure the effectiveness of the thermal data. Bui et al. (2016) compared the output of grayscale images to RGB with different ML classifiers (CNN, SVM, and random forest). It was found that the grayscale CNN outperformed other artificial intelligence classifiers. It was also shown the grayscale CNN had a similar accuracy, using 128 to 300 filters. Zhang et al. (2017) demonstrated the feasibility of CrackNet (a form of CNN). The CrackNet CNN is different than other CNNs, as it uses line filters to enhance the contrast between cracks and the background for preprocessing. The CrackNet also has no pooling layers to remove the downsampling. The study verified that the CrackNet CNN is more effective than SVM and non-ML methods. Further, the pooling layers may not assist in crack detection. However, a comparison of CrackNet CNN to a competing CNN method to demonstrate improvement was not carried out.

Table 2. Comparison of CNN with other methods.

Factors	CNN	NLP	RNN
Parameter-Sharing	Yes	No	Yes
Recurrent Connections	No	No	Yes
Data	Image Data	Tabular Data	Sequence Data (Timeseries, Text, Audio)
Vanishing and Exploding Gradient	Yes	Yes	Yes
Spatial Relationship	Yes	No	No

Fang et al. (2020) combined two CNNs and Bayesian probability to investigate the reduction in the signal-to-noise ratio (SNR). This increased the crack detection but reduced the recall. Simple thresholding (0.5) used in the Bayesian probability was identified as a source of problems. Hence, there is a lack of direct comparison to improve CNNs using preprocessing methods. The comparison of CNN with other methods is demonstrated in Table 2. On comparison with other techniques, it is evident that CNN has parameter-sharing and spatial relationships, along with vanishing and exploding gradients. The exploding and disappearing gradient issues often arise when gradient-based learning methods are used. To address these problems, different approaches, such as ReLU, are used.

2.5. Evaluating Classification

Different evaluation metrics have been used for crack detection. Some examples of this are shown in Table 3. The F1 score and accuracy are the most used methods of evaluating the classification systems for crack detection [7].

Accuracy is the number of correctly predicted instances over the total instances and is a favoured performance metric. However, unbalanced dataset accuracy can still be used as a useful metric, but it can become an unreliable measure of model performance. Many machine learning models are designed around the assumption of balanced class distribution and always predict the majority class. For example, a class with a much larger sample will produce an overoptimistic estimation of the majority class. An unbalanced dataset makes accuracy an unreliable metric [29,30]. Therefore, for imbalanced datasets, it is better to use alternative metrics to summarize model performance.

The F1 score is one of the most widely used metrics in ML studies. This includes both binary and multiclass classification. It is defined as the harmonic mean of precision and recall. The F1 score ranges from zero to one. The minimum value is reached when no positive samples are classified, and the maximum value is reached when there are no false negatives or positives. The F1 score is independent of true negative results [29–32]. Therefore, the true negatives will have a lesser impact on the viability of a crack detection model.

Table 3. Previous studies' methods of crack detection and measurement.

Error	Method	Type	Preprocessing	Reference
1~2% (Crack length/width)	CNN	Deep learning (R-CNN)	None	[1]
<11% length	NiBlack, Sauvola, Wolf, NICK, Bernsen	Image processing	Grayscale	[17]
<10% width	Global analysis + binarization	Image processing	Terrestrial laser scanning (TLS), orthorectification	[3]
mAP 95.54%	CNN	Deep learning (R-CNN)	Thermally excited infrared images	[33]

Table 3. *Cont.*

Error	Method	Type	Preprocessing	Reference
Sensitivity: 93%	Random forest	Traditional machine learning	Binarization	[34]
F1:73–99%	CNN (7 pretrained CNNs)	Deep learning	None	[35]
F1: 91.9%	FCN crack segmentation	Deep learning (VGG16 pretrained)	None	[36]
False discovery rate: 3.86%	Template matching and threshold	Image processing	Convert to 3D with fast average reconstruction	[37]
F1: >87%	CNN (multiscale fusion)	Deep learning (SegNet pretrained)	None	[38]
AUC: 96.8%	Naive Bayes data fusion scheme CNN	Deep learning and traditional machine learning	None	[39]
F1: >80%	CNN	Deep learning (AlexNet pretrained)	Edge detection	[9]
F1: >0.79, length range: 221.82%	FCN crack segmentation	Deep learning (VGG19 pretrained)	None	[33]
F1: 91.7%	FCN pixel detection	Deep learning (VGG16 pretrained)	Pixels annotated (crack)	[27]
F1: 90%	FCN	Deep learning (U-Net pretrained)	Crack-labelled, Adam optimization	[40]
Best F1: 92.6%, others: >72.3%	Faster R-CNN, DCNN, and Bayesian probability	Deep learning (VGG16 and ResNet101) and traditional machine learning	Semiautomatic crack annotation	[41]
F1: 88.86%	CNN	Deep learning (CrackNet CNN)	Line filters (“feature extractor”)	[42]
ACC: >87.9%	CNN	Deep learning (deep CNN)	Image annotation	[43]
Realization of automated system	Agglomerative hierarchical clustering	Traditional machine learning	Removal of distortion, thresholding	[44]
F1: >89%, Pr: >91%,	CNN	Deep learning	Increase ratio of sample (1:3)	[45]
Distance Error: 7.5%–8.5%	Gaussian colour distribution	Traditional machine learning	Particle filtering	[46]
F1: 97%, Pr: 95.5%	Various parametric, nonparametric, clustering, one-class classifiers	Traditional machine learning and image processing	Smoothing, white lane line detection, image normalization, saturation	[18]

The existing research has investigated and established how these preprocessing methods affect IP for crack detection, but not as much on DL. Studies have been focused on novel approaches and combining IP methods to measure the cracks. Studies have rarely determined if preprocessing image methods influence DL classification performance. These methods were explored to improve IP crack detection techniques to minimize noise and complex features and reduce computational cost. This gap was identified by Shahriar and Li (2020) [13,47], who proposed a preprocessing method. The paper proposed using a PASCAL Visual Object Classes Challenge (PASCAL VOC) dataset to determine the appropriate preprocessing method, focusing on denoising. The current study aims to investigate the general preprocessing methods to determine if there is a tangible impact on CNN outputs.

3. Methodology

The method adopted in the current study can be categorized into four stages: study dataset collection, IP, transfer learning, and model analysis. Figure 6 shows the overview of the method used for the study, which is explained subsequently.

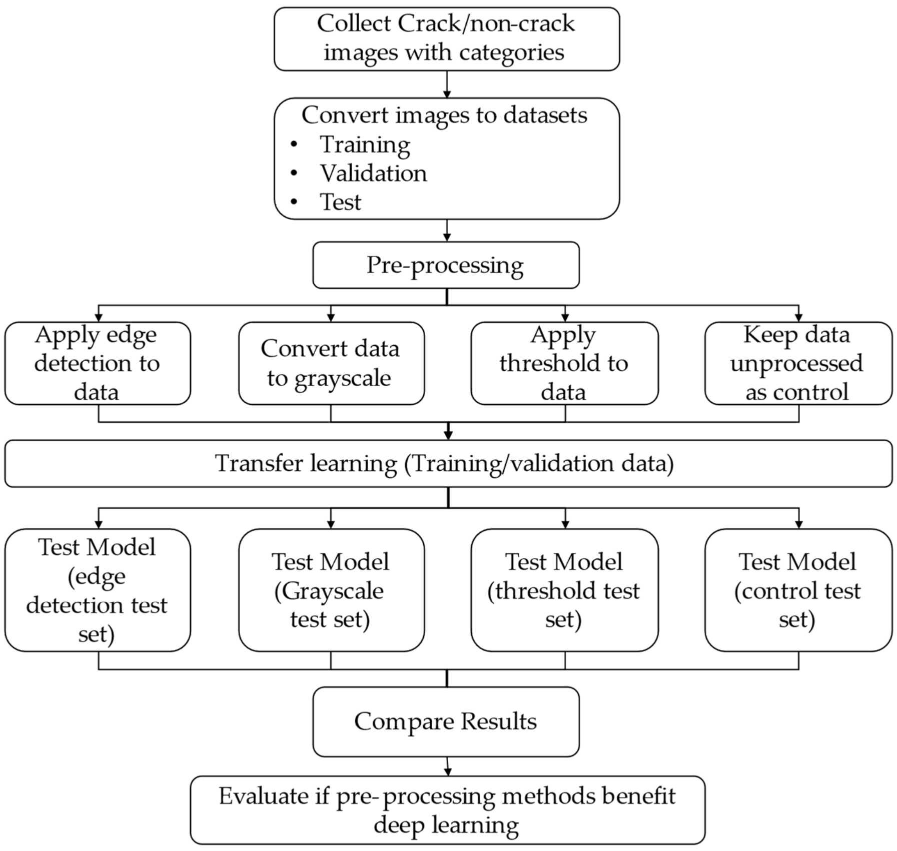

Collect Crack/non-crack images with categories
Convert images to datasets
• Training
• Validation
• Test
Pre-processing
Apply edge detection to data
Convert data to grayscale
Apply threshold to data
Keep data unprocessed as control
Transfer learning (Training/validation data)
Test Model (edge detection test set)
Test Model (Grayscale test set)
Test Model (threshold test set)
Test Model (control test set)
Compare Results
Evaluate if pre-processing methods benefit deep learning

Figure 6. Flowchart of the methodology.

3.1. Dataset Collection

The dataset size and content greatly affect the performance of the models. Smaller images reduce the data size, increasing the training speed. On the other hand, images that are too small do not contain enough data. VGG16 architecture has a default input size of 224 × 224, but can accommodate slightly larger images. The publicly available "Concrete Crack Images for Classification" dataset [48,49] was used for this study. This dataset contains 40,000 images with RGB channels at 227 × 227 pixels. The images were arranged into positive (crack) and negative (non-crack) categories. Each category contains 20,000 images. [35] The Özgenel (2019) dataset images (jpeg) were generated by cropping patches (227 × 227 pixels) from 458 (4032 × 3024 pixels) images of the Middle East Technical University buildings, Turkey.

The images were taken perpendicular to the surface at a one-meter distance. The im-ages contain a variety of surface finishes and illumination conditions. The original dataset was randomly split into training (70%: 28,000), validation (15%: 6000), and test (15%: 6000) datasets, following Özgenela and Sorguç (2018). The training dataset was used to train the model and the validation dataset was used during training to monitor the model learning curve. The test dataset was used to evaluate the model's performance [35]. A balanced number of images was retained in each category. The ratio of positive to negative images has been shown to influence training results. The data need to be evened to reduce the bias towards non-cracks. This occurs as the CNN obtains high accuracy with the one value. With a natural ratio (e.g., 1:65), the network will overestimate the non-crack images. The general ratio was 1:3 (crack:non-crack) [45].

3.2. Image Processing

The images were processed separately once imported using TensorFlow and were converted from images to a sequence of elements. Next, the images were processed using the SciKit Image package in Python 3.8 using a separate code. Crack detection through

IP methods uses various image preprocessing techniques to enhance crack features. This is achieved by simplifying the data (grayscale, thresholding) or removing errant data (smoothing to reduce noise).

DL methods do not typically use image preprocessing techniques, as these reduce the amount of data for the algorithm to learn [12]. The images were processed to obtain four datasets: RGB (control), grayscale (luminance), edge detection (Sobel filter), and binarization (Otsu's method). Figure 7 demonstrates the effect of the IP methods used on crack images.

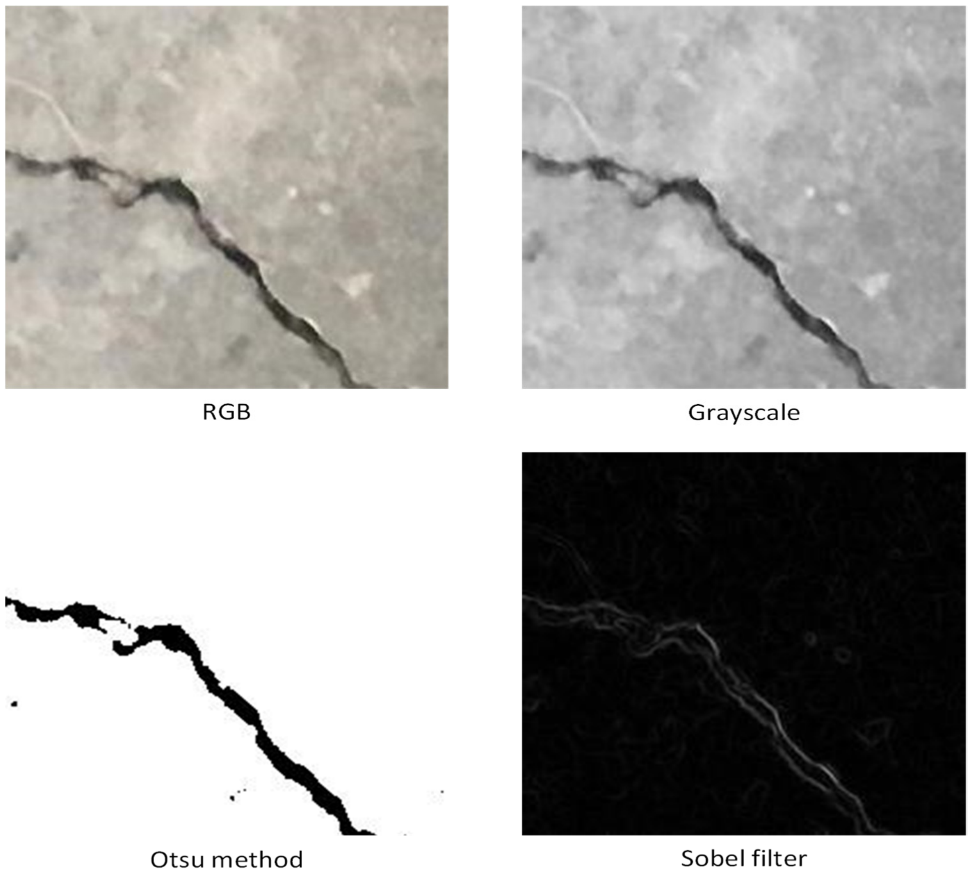


Figure 7. Comparison of IP methods (RGB, grayscale, Otsu method, and Sobel filter) used on a crack image.

3.2.1. Control (RGB)

Standard colour images were used in CNN image categorization. These images featured values for each of the primary colours on the spectrum: red, green, and blue, also known as 3-channels (control RGB). A combination of these three can produce all possible colour pallets. The image is made up of multiple pixels, with every pixel consisting of three different values for the RGB channels. The values of these channels and the channels of the surrounding pixels were used to identify cracks.

3.2.2. Grayscale (Luminance)

Grayscale is the conversion of an RGB image into a brightness scale. This method reduces the amount of data in each image by a third by combining the 3-channels into 1-channel, thereby reducing the image size. This reduction in image size allows for faster training for the algorithm and identification of the cracks in images. The luminous efficiency of each colour (red, green, and blue) is different. Green appears brightest, and blue appears darkest. Poynton (1997) defines the weights to compute true Commission international de l'éclairage (CIE) luminance (Y) on contemporary monitors from linear red (R), green (G), and blue (B) values using Equation (1):

$$Y = 0.2125R + 0.7154G + 0.0721B \tag{1}$$

3.2.3. Edge Detection (Sobel Filter)

The Sobel filter is one of the most widely used IP methods for crack detection [1,9,33,46]. It is a gradient-based method that looks for strong changes in the first derivative of an image. The Sobel edge detector uses a pair of 3 × 3 convolution masks, one estimating the gradient in the x-direction and the other in the y-direction. It is commonly used for both edge detection and eliminating residual noise in IP-based crack detection [5]. The Sobel filter is described using Equations (2)–(4) [9]:

$$E_{i,j} = \sum_{k=1}^{m} \sum_{\mathcal{L}=1}^{n} I(i+k-1, j+\mathcal{L}-1)\ K(k,\mathcal{L}) \tag{2}$$

where

$$E_{i,j} = \text{ sum of the element by element products as a convoluted image}$$

$$E_{i,j}\ dimensions\ are\ (M-m+1) \times (N-n+1)$$

$$I = \text{ image intensity } (dimension\ M \times N)$$

$$K = \frac{kernel}{filter}(dimension\ m \times n)$$

$$K_{Sx} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{3}$$

$$K_{Sy} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & -1 \end{bmatrix} \tag{4}$$

where

$$K_{Sx} = Sobel\ filter\ in\ x\ direction$$

$$K_{Sy} = Sobel\ filter\ in\ y\ direction$$

3.2.4. Thresholding/Binarization (Otsu Method)

Otsu's method is a common thresholding technique used in many IP methods [5,34]. It is a form of automatic image thresholding. This method is based on the idea that an image has two groups: the background image and the target. The Otsu method determines these two groups using a between-class variance. The variance in the brightness distribution is a measure of homogeneity. This determines the threshold value. Thresholding is a method used to enhance crack features for morphological algorithms in IP crack detection [5]. Talab et al. (2016) define the algorithm for Otsu's method using Equation (5) [50]:

$$\sigma_b^2 = Arg\ Max_{0 \le t \le m-1}\left[\omega_0(t)(\mu_0(t)-\mu)^2 + \omega_1(t)(\mu_1(t)-\mu)^2\right] \tag{5}$$

where

$$\sigma_b^2 = Between\ class\ variance$$

$$\mu = Total\ mean = \omega_0(t)\mu_0(t) + \omega_1(t)\mu_1(t)$$

$$t = gray\ value\ threshold$$

$$\omega_0 = Target\ pixel\ ratio\ in\ image$$

$$\mu_0 = Target\ mean$$

$$\omega_1 = Background\ pixel\ ratio\ in\ image$$

$$\mu_1 = Background\ mean$$

3.3. The Proposed CNN Model

The current study was performed using pretrained CNN architecture. Transfer learning with pretrained architecture has proven to increase the efficiency and accuracy of crack classifiers [9]. Furthermore, VGG16 is the most widely used pretrained architecture in crack detection and on ImageNet [36,51].

The proposed CNN architecture consists of convolutional blocks and a fully connected layer (see Figure 8). Each convolutional block consists of a convolutional layer, activation unit, and pooling layer. The convolutional layer performs a convolution operation on the output of the previous layer using kernels/filters [36,52,53]. The kernel is a matrix of weights used across the image to extract classification features. In a CNN model, the data have the spatial structure $Xl \in (Hl \times Wl \times Cl)$. This is a 3D array or tensor. The spatial dimensions are height (H) and width (W). The third dimension (C) is the number of feature channels. (x) is an im2row operator extracting $W' \times H'$ patches from map x, storing them as rows of $(H\text{ “}W\text{”}) \times (H'W'D)$. Vedaldi et al. (2015) provide the convolutional matrix formula using Equation (6):

$$vec\ y = vec\ (\phi(x)F), \frac{dz}{dF} = \phi(x)^T \frac{dz}{dY}, \frac{dz}{dX} = \phi\left(\frac{dz}{dX}F^T\right) \tag{6}$$

where

$$F \in \mathbb{R}^{(H'W'D)\times K}$$

$$Y \in \mathbb{R}^{(H\text{ “}W\text{”})\times K}$$

$$F \in \mathbb{R}^{(HW)\times D}$$

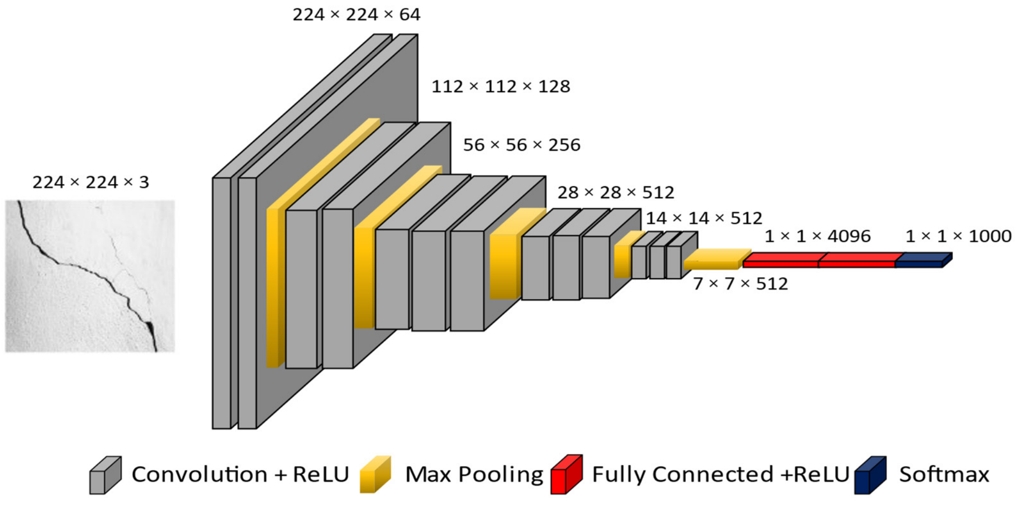


Figure 8. Original network architecture of VGG16 for image classification.

The activation unit is a function that determines if the block will be activated. The most common function used is the rectified linear unit (ReLU). Vedaldi et al. (2015)'s guide to CNNs in matrices states the ReLU function using Equation (7):

$$vec\ y = diag\ s\ vec\ x, \frac{dz}{d\ vec\ x} = diag\ s \frac{dz}{d\ vec\ y} \tag{7}$$

where

$$s = [vec\ x > 0] \in \{0,1\}^{HWD}\ (indicator\ vector)$$

The pooling layer reduces the number of parameters and computation through down-sampling. There are two methods used for pooling: max pooling and average pooling. Max pooling is more commonly presented in Equation (8) (Vedaldi et al. (2015)):

$$vec\ y = S(x)\ vecx, \frac{dz}{d\ vec\ x} = S(x)^T \frac{dz}{d\ vec\ y} \tag{8}$$

where

$$Matrix\ S(x) \in \{0,1\}^{(H\text{ “}W\text{” } D)x(HWD)}$$

The final sections are a fully connected layer and a softmax layer. The fully connected layer is connected to all previous layers' activations. The activations are computed with matrix multiplication, followed by a bias offset [54]. The loss function is a method to correct the difference between the predicted and true outcomes in training. The most common loss function is Softmax [55]. Vedaldi et al., (2015) define the softmax function using Equation (9):

$$\frac{dz}{dX} = Y \odot \left(\frac{dz}{dY} - \left(\frac{dz}{dY} \odot Y\right) 11^T\right) \tag{9}$$

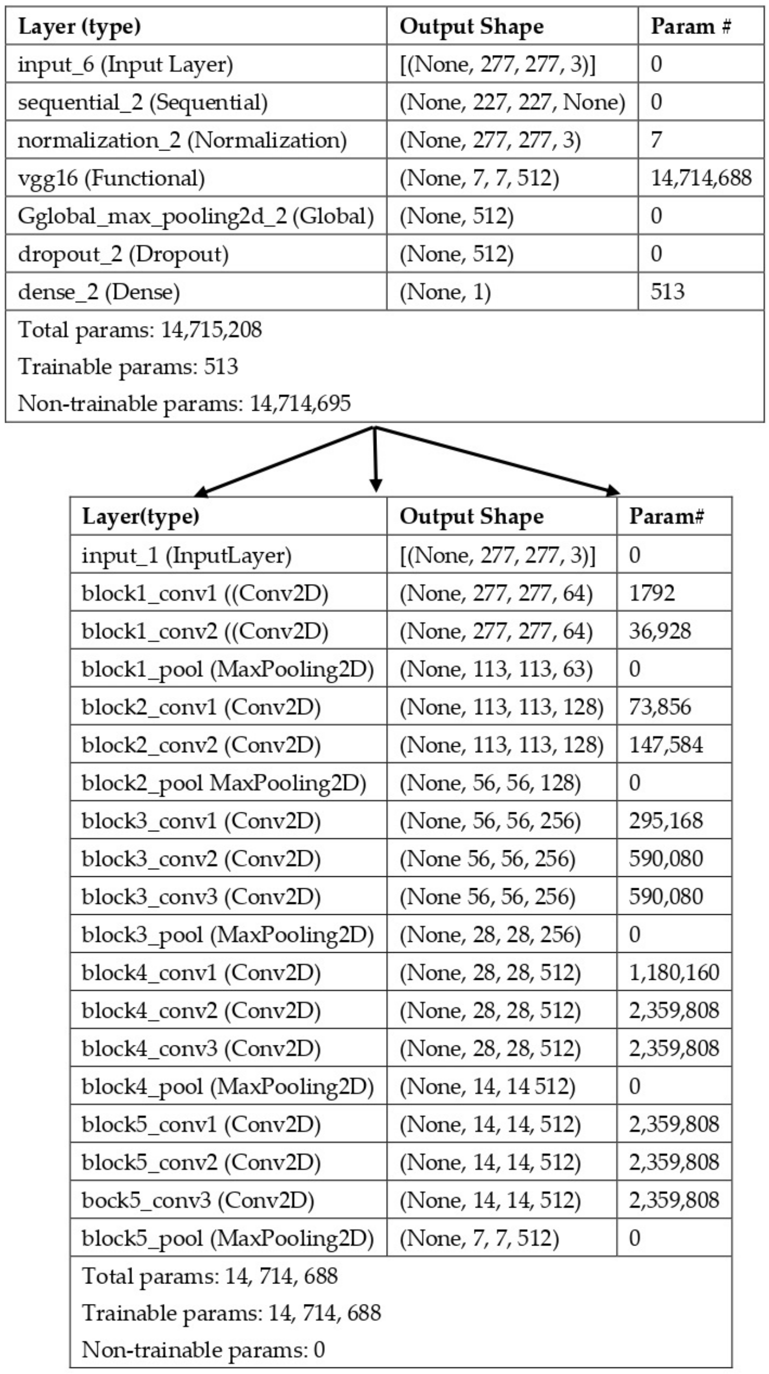
3.3.1. Model Development

The computing was performed on a laptop (RAM: 16GB, CPU: AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz, GPU: NVIDIA GeForce GTX 1650 Ti, 64-bit operating system) to develop the model. The code was programmed using Anaconda Spyder 4.1.5 (Python 3.8) with Keras 2.4 and TensorFlow 2.5 python packages. This allowed any system with Python and the appropriate PIPs or Python Notebook to run the CNN.

The Keras package contains many DL architectures. The Keras VGG16 model is based on the model proposed by Simonyan and Zisserman (2014) [56]. Max pooling was used to achieve the best results in crack detection [57]. The model was trained using an Adam optimizer and binary cross-entropy for loss [35]. The training epochs were also determined. One epoch trained the model on all images, with each step being one batch. The recommended batch size of 32 images was used. Two training epochs were used: 10-epochs has previously approached convergence with RGB images [35], and 20-epochs to allow more time to compensate for single-channel images.

The model contained several layers, as shown in Figure 9. The input layer defined the input parameters for the model: location, labels, number of channels, the dataset's name, and size (pixels). The sequential layer provided the data augmentation. The data were augmented using a rotation ($0.2/2\pi$) and horizontal and vertical flipping to increase data size and remove directional bias. Finally, the normalization layer normalized the data (from 0–255) to allow the use of ImageNet weights. The ImageNet weights used for pretraining were from -1 to +1. The VGG16 architecture was then used in pretraining.

The pooling layer converted the base model output shape to vectors. The dropout function dropped out filters during training to regulate the results. Finally, the dense classifier layer combined the models' values into a binary result. This result was used to predict the label of the crack. A non-crack was labelled 0, while a crack was labelled 1. The program rounded down to predict a crack; a function was created to convert numbers $\geq$0.5 to 1. This allowed the program to determine if an image was a crack or not.



Layer (type)	Output Shape	Param #
input_6 (Input Layer)	[(None, 277, 277, 3)]	0
sequential_2 (Sequential)	(None, 227, 227, None)	0
normalization_2 (Normalization)	(None, 277, 277, 3)	7
vgg16 (Functional)	(None, 7, 7, 512)	14,714,688
Gglobal_max_pooling2d_2 (Global)	(None, 512)	0
dropout_2 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 1)	513

Total params: 14,715,208
Trainable params: 513
Non-trainable params: 14,714,695

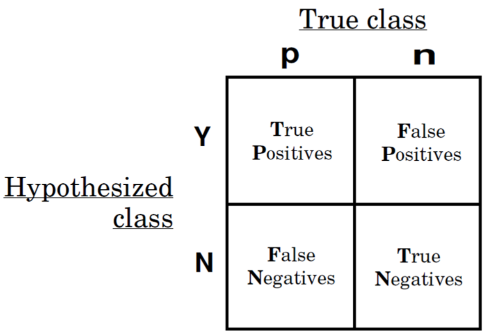
Layer(type)	Output Shape	Param#
input_1 (InputLayer)	[(None, 277, 277, 3)]	0
block1_conv1 ((Conv2D)	(None, 277, 277, 64)	1792
block1_conv2 ((Conv2D)	(None, 277, 277, 64)	36,928
block1_pool (MaxPooling2D)	(None, 113, 113, 63)	0
block2_conv1 (Conv2D)	(None, 113, 113, 128)	73,856
block2_conv2 (Conv2D)	(None, 113, 113, 128)	147,584
block2_pool MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
bock5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0

Total params: 14, 714, 688
Trainable params: 14, 714, 688
Non-trainable params: 0

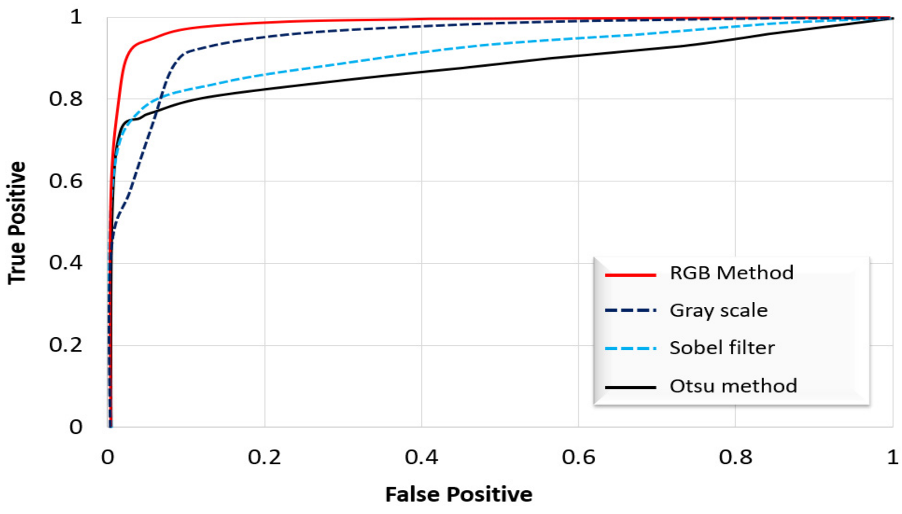
Figure 9. RGB model used as displayed by Python; the VGG16 model layer has been expanded to show the CNN.

3.3.2. Model Analysis

In analysing the model, the classifiers were evaluated on their ability to predict classes. In each instance, the data had a class label, either positive or negative (p, n). The classification model predicted a class as yes or no (Y, N). A classifier with two discrete classes produced four results for each instance. These results are summarized on a confusion matrix (see Figure 10a) [58].



(**a**)



(**b**)

Figure 10. (**a**) Confusion matrix. (**b**) Comparison of four approaches with ROC curves.

Whereas a false positive relates to values incorrectly predicted an actual positive, i.e., negative values predicted as positive, false negatives relate to positive values predicted as negative, while true positive and true negative are values that are predicted correctly. The classification performance was evaluated using five evaluation criteria: accuracy (ACC), true positive rate (TPR), true negative rate (TNR), positive predictive value (PPV), negative predictive value (NPV), and F1 score. The F1 score is the harmonic mean of the data. Dorafshan et al. (2018) state the formula to evaluate the classification based on the mentioned criteria using Equations (10)–(15):

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{10}$$

$$TPR = \frac{TP}{TP + FN} \tag{11}$$

$$TNR = \frac{TN}{FP + TN} \tag{12}$$

$$PPV = \frac{TP}{TP + FP} \tag{13}$$

$$NPV = \frac{TN}{TN + FN} \tag{14}$$

$$\mathrm{F1} = \frac{2TP}{2TP + FN + FP} \tag{15}$$

The formulas were all created using the same four values: true positive (TP), true negative (TN), false positive (FP), and false negative (FN) [23]. This allowed the CNN to be properly evaluated based on the classifications.

Recall (TPR and TNR) is the ability of the CNN to find all relevant category cases within the data. Precision (PPV and NPV) is the ability of the CNN to identify only the correct category. The F1 score is preferred, as it reduces the effect of extreme results in either precision or recall, being the harmonic mean of both. The results of all models were compared to the control (RGB) to determine if there was any improvement, as subsequently presented. Figure 10b shows the difference between ROC curves in four approaches. The ROC curve of the grayscale method was higher than the others.

4. Results

Different CNN models were created using transfer learning to generate eight models (four types of images at two training times). The models produced confusion matrices, allowing the performance to be compared and evaluated. The results of the confusion matrices generated using the test data (6000 sample images) and the derived metrics are explained in this section. The validation data were used to train the model. The model training accuracy was plotted at each epoch to visualize the training progress and depict the model improvement during training (see Figure 11). The models showed stark improvement in the first two epochs, followed by smaller increments.

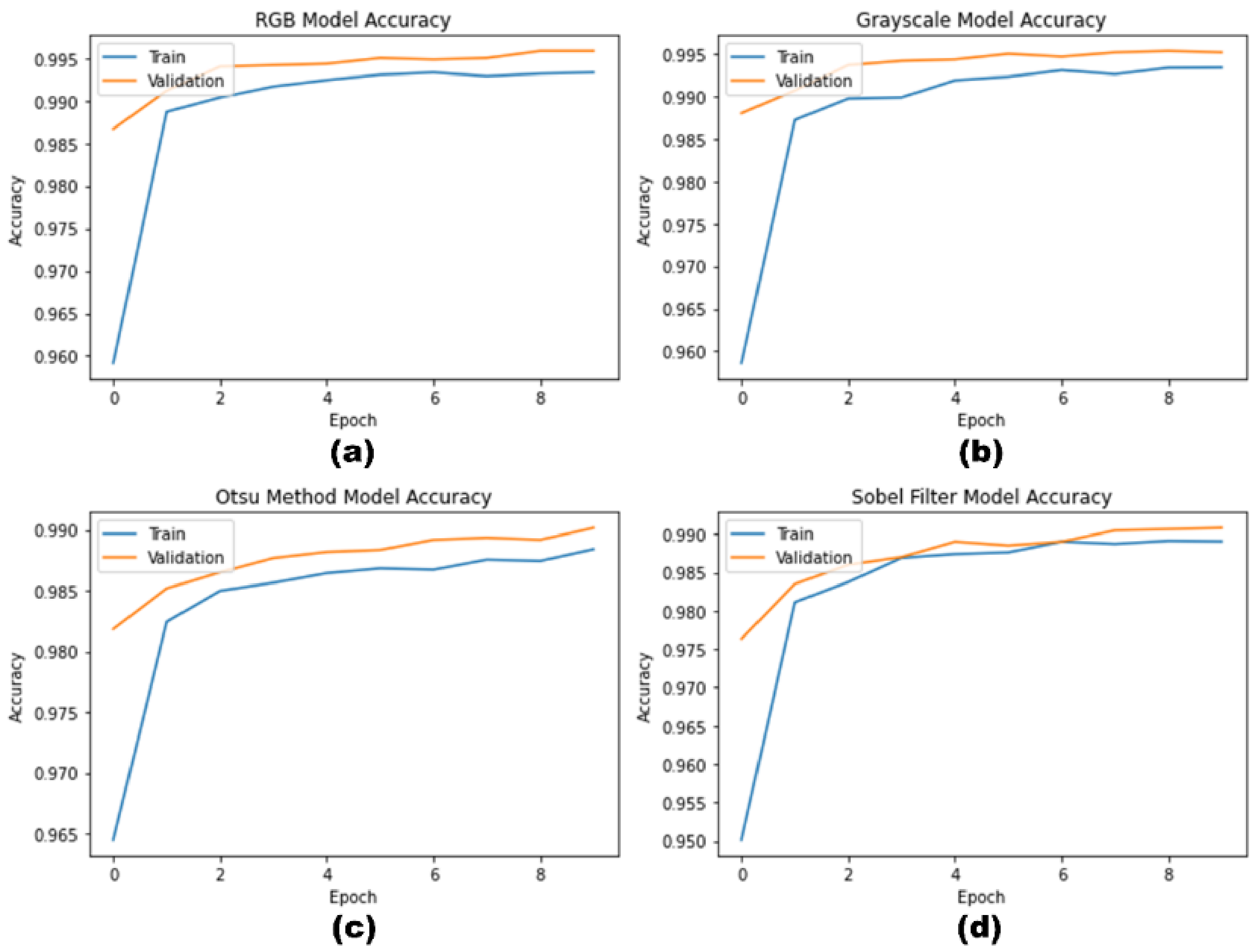


Figure 11. (**a**–**d**): The 10-epoch model training accuracy at each epoch demonstrates model accuracy at each training epoch.

The RGB data were selected as the control for testing purposes. However, the pre-trained VGG16 model only had access to pretrained RGB weights. This was exaggerated as the RGB weights were designed for three channels (red, green, blue), while the processed images had only one (brightness). Therefore, the RGB results were expected to improve performance, depending on the model's reliance on colour.

4.1. 10-Epoch Training

The initial objective of the study was to determine if processing images could improve the performance of DL. For this purpose, the initial test was performed using 10 epochs of training. It was decided to train the data for 10 epochs as this is sufficient for convergence with many pretrained networks [35].

The confusion matrices (see Figure 12) provide a full picture of the four models' performance. The results showed high performance with an accuracy greater than 98% in all tests. The largest individual error in the results was 49 out of 3000 (1.63% Otsu method crack imagery) incorrectly labelled images. The best individual result was 10 out of 3000 (0.33% grayscale non-crack imagery) incorrectly labelled images. In general, the results showed that the model had over twice as many incorrect predictions for the crack images compared to the non-crack images.

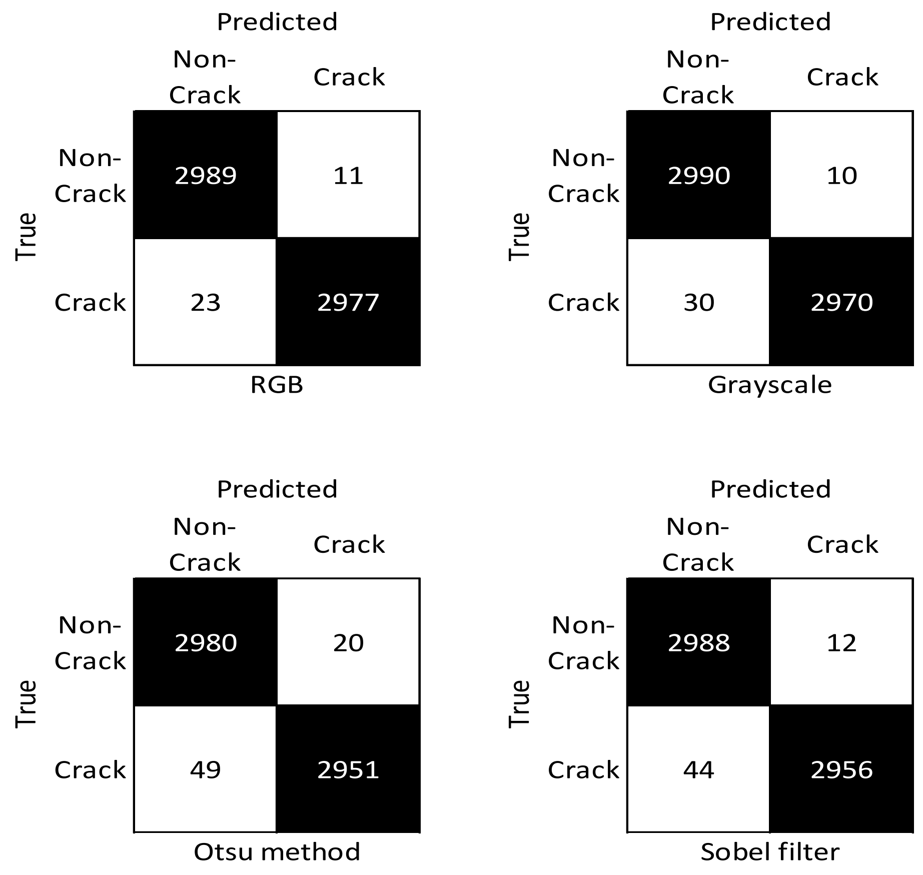


Figure 12. Confusion matrix results of the model for each image processing method run for 10 epochs.

The grayscale results were similar to the RGB, with a minor increase in false negative results. The Sobel filter returned a comparable number of false positives (12:11) but nearly double the false negatives (44:23) of the control (RGB). The Otsu method returned nearly double the false positives (20:11) and over double the false negatives (49:23) compared to RGB. Overall, as shown in Figure 12, the RGB and grayscale were similar, with the Sobel filter increasing the number of false negatives, and the Otsu method reducing performance in all areas.

Table 4 depicts that the control RGB performed the best, achieving over 99% in all metrics. This was expected, as the pretrained weights were created using an RGB dataset. The accuracy of the models for each IP method varied from 98.85% to 99.43%. This showed only a minor reduction in any IP method compared to the control. The grayscale model was the best-processed image (−0.10% from RGB). The F1 score was used to evaluate the model's accuracy in predicting a crack. The RGB image model performed best, while the grayscale model's F1 score was slightly lower (−0.10%). Overall, the RGB and grayscale models achieved similar results, while the Otsu method returned the worst performance of all metrics, though the difference was minor.

Table 4. Metrics using confusion matrix results for each image processing method run for 10 epochs.

Metric	**10-Epoch Pretrained**				**Difference to RGB**		
	RGB	**Grayscale**	**Otsu Method**	**Sobel Filter**	**Grayscale**	**Otsu Method**	**Sobel Filter**
ACC	99.433%	99.333%	98.850%	99.067%	−0.100%	−0.583%	−0.367%
TRP	99.233%	99.000%	98.367%	98.533%	−0.233%	−0.867%	−0.700%
TNR	99.633%	99.667%	99.333%	99.600%	0.033%	−0.300%	−0.033%
PPV	99.632%	99.664%	99.327%	99.596%	0.033%	−0.305%	−0.036%
NPV	99.236%	99.007%	98.382%	98.549%	−0.230%	−0.854%	−0.688%
F1	99.432%	99.331%	98.844%	99.062%	−0.101%	−0.588%	−0.371%

The grayscale model outperformed the RGB model in TNR and precision/PPV. The grayscale model was best at finding all non-crack images. The grayscale model also returned the highest percentage (99.664%) of true cracks out of all predicted cracks. The differences between RGB and grayscale were minor, suggesting that colour is not critical in crack detection.

4.2. 20-Epoch Training

The 20-epoch training was performed to determine if greater training would allow the model to improve its compensation for single-channel images (see Figure 13).

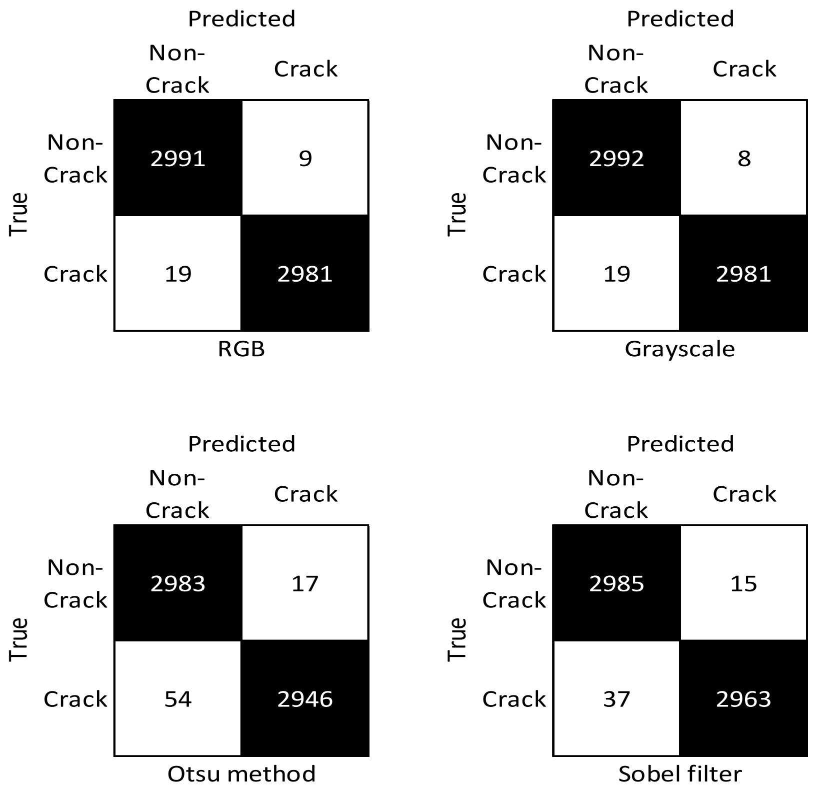


Figure 13. Confusion matrix test results of the model for each image processing method run for 20 epochs.

The confusion matrix results (see Table 5) illustrate that the 20-epoch models had increased performance across all models. The grayscale and RGB were nearly identical (one-image difference). The grayscale model at 20 epochs had the best performance. The RGB model differed from the grayscale by incorrectly labelling one more image as a crack. This indicates that the grayscale model and RGB model evaluated cracks similarly.

Table 5. Metrics using confusion matrix test results for each image processing method run for 20 epochs.

Metric	20-Epoch Pretrained				Difference to RGB		
	RGB	Grayscale	Otsu Method	Sobel Filter	Grayscale	Otsu Method	Sobel Filter
ACC	99.533%	99.550%	98.817%	99.133%	0.017%	−0.717%	−0.400%
TRP	99.367%	99.367%	98.200%	98.767%	0.000%	−1.167%	−0.600%
TNR	99.700%	99.733%	99.433%	99.500%	0.033%	−0.267%	−0.200%
PPV	99.699%	99.732%	99.426%	99.496%	0.033%	−0.273	−0.203%
NPV	99.369%	99.369%	98.222%	98.776%	0.000%	−1.147%	−0.593%
F1	99.533%	99.549%	98.809%	99.130%	0.017%	−0.723%	−0.402%

The Otsu method had the worst performance in this test as well. The Otsu method had 17 more incorrect non-crack labels than the Sobel filter model and 35 more than the RGB or grayscale models. The Sobel filter model, for comparison, had 37 incorrect non-crack labels. The Sobel filter and Otsu method models showed a comparable number of incorrect crack labels. This suggests that the models may have struggled to label the same non-crack images.

The grayscale model outperformed the RGB control in all metrics by less than 0.1% (see Table 5). The remaining IP models performed worse than the RGB control. The worst-performing model was the Otsu method, with an accuracy of 98.817% and an F1 score of 98.809%. The Otsu method model recall/TPR was 1.17% lower than the RGB model. The NPV or precision for the non-cracks also performed poorly, with a 1.15% reduction compared to RGB. Overall, the results were still very good, with all model metrics above 98%.

4.3. Comparison of the Epochs

A comparison between the 20-epoch models and 10-epoch models was performed to determine if the DL compensated for the single-channel images.

The confusion matrices for difference (see Figure 14) showed improvement or deterioration between the models based on their training times. An improvement depicts an increase in the correct predictions (black squares) and a reduction in the incorrect predictions (white squares). It should be noted that changes are cancelled out across the true la-bels (rows). Overall, the largest improvement was achieved in the grayscale model. This model correctly labelled 11 more cracks. The non-crack improvement was identical for both the grayscale and RGB models. The Sobel filter model increased its crack labelling by seven images in the 20-epoch model. The RGB model increased the correct crack image labels by four in the 20-epoch model. This suggests that the one-channel images using three-channel weights converged slower.

Surprisingly, the Otsu method model's incorrect crack-image labelling increased by five. This indicated that the Otsu method images might have more noise, leading to early overfitting. This could also be due to the binarized images exaggerating the brightness changes and making non-crack images with details, such as colour or ripples, appear as cracks. The Sobel filter method showed a decrease in correct non-crack labels, which could be due to the filter creating outlines of image features that possibly appear like cracks. These changes were minor, but the Otsu and Sobel models' performances deteriorated with further training.

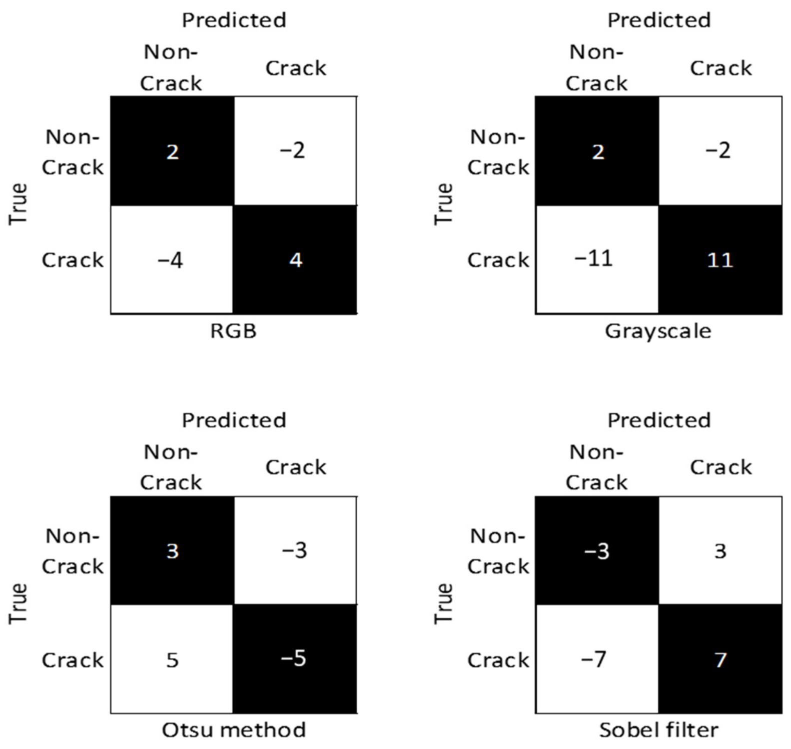


Figure 14. Confusion matrices showing the differences from the 10-epoch models to the 20-epoch models.

The metrics in Table 6 depict the relative improvement between the 10-epoch and 20-epoch models. It further shows the comparative improvement between the IP models. Overall, the grayscale model results showed the most improvement. Contrastingly, the Otsu method and Sobel filter changed in different ways. The Otsu method reduced the NPV and TPR, while the TNR and PPV increased. This showed that the threshold CNN models removed crack features as the recall decreased. The increase in performance of detecting non-cracks led to an increase in precision. While the Sobel filter model showed the opposite effect, it reduced TNR and PPV, while the NPV and TPR increased. This indicated that the edge detection process may have added features in non-crack images, and the CNN model could not differentiate between the cracks. Overall, the models only marginally increased or decreased by any metric. The changes from 10 epochs to 20 epochs are all within ± 0.4%, confirming that convergence is approximately 10 epochs.

Table 6. Model metrics changes from 10 epochs to 20 epochs.

Metric	**Difference from 10 to 20 Epochs**				**Increase Compared to RGB**		
	RGB	**Grayscale**	**Otsu Method**	**Sobel Filter**	**Grayscale**	**Otsu Method**	**Sobel Filter**
ACC	0.100%	0.217%	−0.33%	0.067%	0.117%	−0.133%	−0.033%
TRP	0.133%	0.367%	−0.167%	0.233%	0.233%	−0.300%	0.100%
TNR	0.067%	0.067%	0.100%	−0.100%	0.000%	0.033%	−0.167%
PPV	0.067%	0.068%	0.099%	−0.099%	0.001%	0.032%	−0.167%
NPV	0.132%	0.362%	−0.160%	0.227%	0.230%	−0.293%	0.094%
F1	0.100%	0.218%	−0.035%	0.068%	0.118%	−0.135%	−0.032%

The cracks on the images are shown in the Figure 15a–m. Figure 15a represents the input and output images of the cracks on the concrete structures, whereas Figure 15d,g,j represent the crack input images used to obtain the output of Figure 15f,i,l, respectively, while Figure 15b,e,h,k are labelled images. The input and output images for corrosion are presented in Figure 15a–l, where the left panel corresponds to input, and the right panel corresponds to the output images.

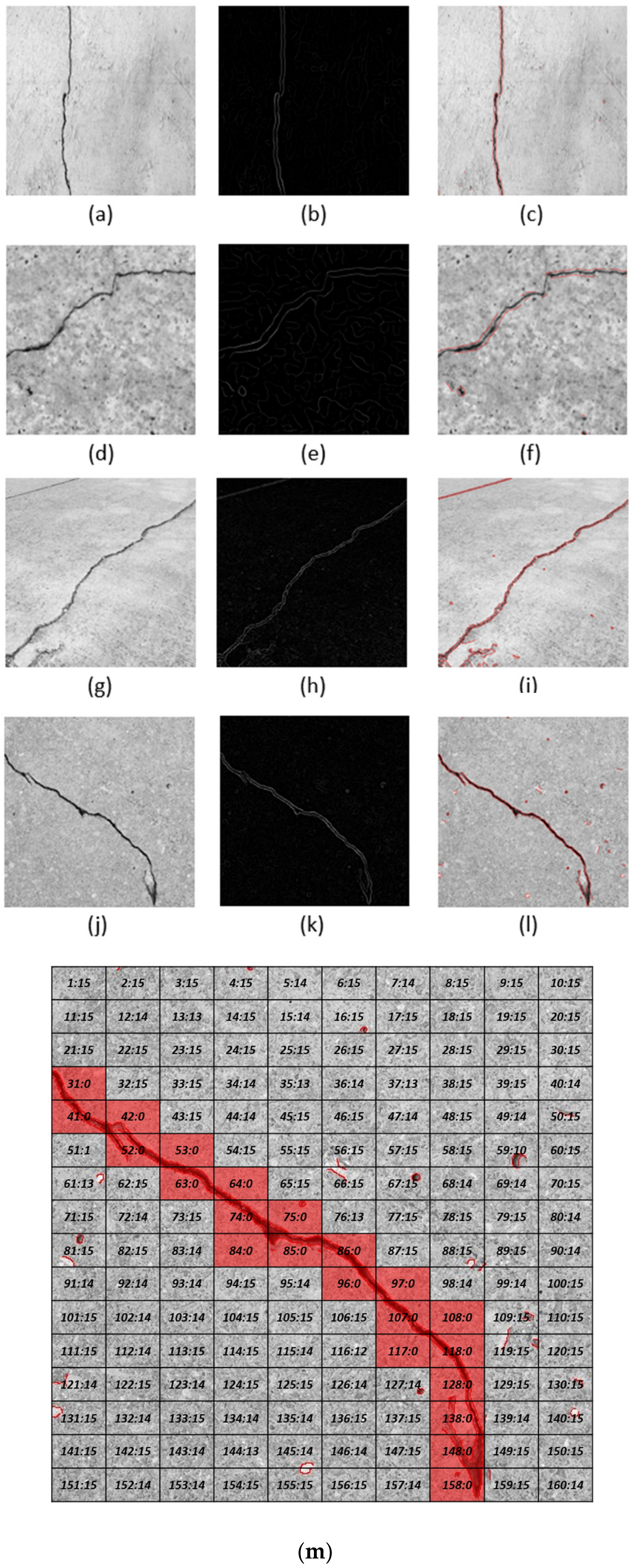


Figure 15. (**a**–**m**): Crack images and associated segmentation in the dataset.

The total crack pixels were classified into significant and weak pixels based on pixel width. Despite other structures in the images, the crack prediction provided a high score for the area where positive classes were present. The cracks with a pixel depth of 0 to 5 were categorized as significant/deep cracks, while those above 5 were classified as weak cracks or object features (see Figure 15). Moreover, a data augmentation approach based on random image cropping and patching was used for the proposed CNN architecture for label generation and crack detection during training. The frequency of crack pixels was predicted, and bounding boxes were located using data distribution and spatial location.

5. Discussion

The RGB models were expected to outperform all other models as the pretrained weights were only available as three channels (RGB). The processed images had only one channel (brightness). All models performed to a very high accuracy (>98%) and F1 score (>98%). There was a marginal change in performance because of increasing the training from 10 to 20 epochs (within ±0.4%). Previously published pretrained models had similar results. Dung and Anh (2019) [36] achieved an accuracy and F1 score of 99.9%. Özgenela and Sorguç (2018) used a similar dataset (RGB), and the same number of epochs achieved an accuracy of 99.9% and an F1 score of 100%. These were marginally better than the current study, which achieved 99.4% for both accuracy and F1 score.

The pretrained model results outperformed CNN models that were trained from scratch. CNN models trained from scratch, such as those of [43], only achieved an accuracy of around 90%. This was expected, as pretrained models have weights generated from millions of images, while models trained from scratch have only thousands. Surprisingly the grayscale CNN models produced the same results as the RGB models. The 20-epoch grayscale model was nearly identical. This suggested that the models do not identify cracks based on colour features. The grayscale results were hard to compare due to the lack of studies on the grayscale CNNs used for crack detection.

For comparison, Bui et al. (2016) [12] compared the performance of RGB (80.8%) and grayscale (82.2%) CNN models for general object classification by training their models from scratch. Similarly, Xie and Richmond (2018) [11] showed that grayscale was 0.5% less accurate than training a model from scratch using ImageNet for general images. These two pretrained models were then used to evaluate test images. The models showed that the grayscale model was faster and outperformed the classification model's accuracy (RGB: 74.98%, grayscale: 77.06%). However, the study used a different CNN architecture (Incep-tionV3). Further, the RGB images used were pseudo-colour from grayscale images, reducing the performance.

Overall, there is a lack of literature on the effect of IP on DL crack detection. The IP methods of automatic crack detection use many techniques, such as thresholding and edge detection, to help the algorithms detect the cracks within the image. The current techniques using DL have focused on more data. The alternate theory is that preprocessing the images could reduce unnecessary information and enhance relevant features [12]. The literature using IP in CNNs compared the grayscale image to RGB but did not use crack images. Bui et al. (2016) [12] used a random image dataset, and Xie and Richmond (2018) [11] used a diseased lung image dataset. Both found that the grayscale technique increased the speed and performance of DL models.

The Otsu method and Sobel filter models performed comparatively poorly at identifying the cracks. These models also performed worse than the control at identifying non-crack images. The further training of these models showed surprising changes. The Otsu method increased the detection of true negative images and decreased the true positive images. The Sobel filter, conversely, showed an increased detection of true positive images and decreased true negatives. This suggests that CNN models created with binarized im-ages detect fewer true positives in crack detection. The results for the Sobel filter model suggest that true negatives are harder to identify when using edge detection. Both the Otsu method and Sobel filter models outperformed the other models. This implies that binarization and edge detection may remove relevant features and information used in CNN crack classification. The grayscale and RGB models produced similar results, suggesting that colour is not critical in crack detection. The model was created using pretrained weights; further investigation for comparing models "trained from scratch" could confirm the results.

6. Conclusions

The study investigated the effects of preprocessing images on the performance of DL crack detection using a dataset of 40,000 images. The results depicted that using a pretrained model with RGB weights and grayscale images does not affect the performance of a CNN model for detecting cracks in the concrete structure. The other IP methods (thresholding and edge detection) reduced the performance. The grayscale was found to be promising in reducing the noise of the image without removing relevant features. The study used the Keras Python package and pretrained VGG16 to develop the CNN. The original image dataset was converted using the SciKit Image Python package into four sets to compare: RGB (control), luminance (grayscale), Otsu method (thresholding), and Sobel filter (edge detection).

The grayscale models (F1 score for 10 epochs: 99.331%, 20 epochs: 99.549%) demonstrated a similar performance to the RGB models (F1 score for 10 epochs: 99.432%, 20 epochs: 99.533%). This suggests the features used in DL to identify cracks do not rely on colour. The edge detection models and thresholding models performed worse. The edge detection models showed that the model became better at detecting cracks (+7 images) and worse at detecting non-cracks (−3 images) with more training. Conversely, the thresholding models showed that the model became better at detecting non-cracks (+3 images) and worse at detecting cracks (−5 images) with more training. The models performed very well with accuracies and F1 scores (all above 98%).

This study demonstrated that colour is not a relevant feature in DL crack detection. This was promising, as colour images are larger, and decreasing image data size could increase processing speed and decrease the data size needed for storage. These results may be misleading due to the RGB weights in the pretrained model. Nevertheless, this demonstrates that grayscale images can improve performance. However, further studies should investigate these results using fully trained and segmented models. The testing, results, and analysis were performed on a pretrained model. The weights for the pretrained models are only available in three-channel (RGB). This was performed due to a lack of time and the greater knowledge needed to execute. Further research should either train the models from scratch on their respective images or obtain one-channel (grayscale) pretrained weights. Further research could be carried out on the effects of IP FCN pixel segmentation to increase pixel accuracy. In the future, thresholds of 106 and 101 could be evaluated to investigate the effect of threshold in crack detection. Moreover, other pretrained models, such as VGC11, VGC19, and AlexNet, could be investigated to compare their performance.

Author Contributions: Conceptualization, V.P.G. and H.S.M.; methodology, V.P.G. and Z.G.; software, V.P.G.; validation, V.P.G., H.S.M. and Z.G.; formal analysis, V.P.G.; investigation, H.S.M.; resources, V.P.G.; data curation, V.P.G.; writing—original draft preparation, H.S.M.; writing—review and editing, V.P.G., Z.G., H.S.M. and F.U.; visualization, V.P.G. and H.S.M.; supervision, Z.G.; project administration, Z.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Codes are available and will be provided upon reasonable request to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kim, I.-H.; Jeon, H.; Baetk, S.-C.; Hong, W.-H.; Jung, H.-J. Application of crack identification techniques for an aging concrete bridge inspection using an unmanned aerial vehicle. *Sensors* **2018**, *18*, 1881. [CrossRef] [PubMed]
2. Munawar, H.S.; Aggarwal, R.; Qadir, Z.; Khan, S.; Kouzani, A.; Malhmud, M. A gabor filter-based protocol for automated image-based building detection. *Buildings* **2021**, *11*, 302. [CrossRef]
3. Valença, J.; Puente, I.; Júlio, E.; González-Jorge, H.; Alrias-Sánchez, P. Assessment of cracks on concrete bridges using image processing supported by laser scanning survey. *Constr. Build. Mater.* **2017**, *146*, 668–678. [CrossRef]
4. Munawar, H.S.; Khan, S.I.; Qadir, Z.; Kiani, Y.S.; Kouzani, A.Z.; Mahmud, M.A.P. Insights into the Mobility Pattern of Australians during COVID-19. *Sustainability* **2021**, *13*, 9611. [CrossRef]
5. Mohan, A.; Poobal, S. Crack detection using image processing: A critical review and analysis. *Alex. Eng. J.* **2018**, *57*, 787–798. [CrossRef]
6. Munawar, H.S.; Khan, S.; Qadir, Z.; Kouzani, A.; Mahmud, M. Insight into the impact of COVID-19 on Australian transportation sector: An economic and community-based perspective. *Sustainability* **2021**, *13*, 1276. [CrossRef]
7. Hsieh, Y.-A.; Tsai, Y.J. Machine learning for crack detection: Review and model performance comparison. *J. Comput. Civ. Eng.* **2020**, *34*, 04020038. [CrossRef]
8. Khan, S.I.; Qadir, Z.; Munawar, H.S.; Nayak, S.R.; Budati, A.K.; Verma, K.; Prakash, D. UAVs path planning architecture for effective medical emergency response in future networks. *Phys. Commun.* **2021**, *47*, 101337. [CrossRef]
9. Dorafshan, S.; Thomas, R.J.; Maguire, M. Benchmarking image processing algorithms for unmanned aerial system-assisted crack detection in concrete structures. *Infrastructures* **2019**, *4*, 19. [CrossRef]
10. Liaquat, M.U.; Munawar, H.S.; Rahman, A.; Qadir, Z.; Kouzani, A.Z.; Mahmud, M.A.P. Sound localization for ad-hoc microphone arrays. *Energies* **2021**, *14*, 3446. [CrossRef]
11. Xie, Y.; Richmond, D. Pre-training on grayscale imagenet improves medical image classification. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, Munich, Germany, 8–14 September 2018.
12. Bui, H.M.; Lech, M.; Cheng, E.; Neville, K.; Burnett, I.S. *Using Grayscale Images for Object Recognition with Convolutional-Recursive Neural Network*; IEEE: Piscataway, NJ, USA, 2016.
13. Shahriar, M.T.; Li, H. A Study of Image Pre-processing for Faster Object Recognition. *arXiv* **2020**, arXiv:Preprint/2011.06928.
14. Pranno, A.; Greco, F.; Lonetti, P.; Luciano, R.; De Maio, U. An improved fracture approach to investigate the degradation of vibration characteristics for reinforced concrete beams under progressive damage. *Int. J. Fatigue* **2022**, *163*, 107032. [CrossRef]
15. De Maio, U.; Greco, F.; Leonetti, L.; Blasi, P.N.; Pranno, A. A cohesive fracture model for predicting crack spacing and crack width in reinforced concrete structures. *Eng. Fail. Anal.* **2022**, *139*, 106452. [CrossRef]
16. Wang, G.; Peter, W.T.; Yuan, M. Automatic internal crack detection from a sequence of infrared images with a triple-threshold Canny edge detector. *Meas. Sci. Technol.* **2018**, *29*, 025403. [CrossRef]
17. Kim, H.; Ahn, E.; Cho, S.; Shin, M.; Sim, S.-H. Comparative analysis of image binarization methods for crack identification in concrete structures. *Cem. Concr. Res.* **2017**, *99*, 53–61. [CrossRef]
18. Oliveira, H.; Correia, P.L. CrackIT An image processing toolbox for crack detection and characterization. In Proceedings of the 2014 IEEE International Conference on Image Processing (ICIP), Paris, France, 27–30 October 2014; pp. 798–802.
19. Akram, J.; Munawar, H.S.; Kouzani, A.Z.; Mahmud, M.A.P. Using Adaptive Sensors for Optimised Target Coverage in Wireless Sensor Networks. *Sensors* **2022**, *22*, 1083. [CrossRef]
20. Akram, J.; Tahir, A.; Munawar, H.S.; Akram, A.; Kouzani, A.Z.; Mahmud, M.A.P. Cloud-and Fog-Integrated Smart Grid Model for Efficient Resource Utilisation. *Sensors* **2021**, *21*, 7846. [CrossRef]
21. Qadir, Z.; Munir, A.; Ashfaq, T.; Munawar, H.S.; Khan, M.A.; Le, K. A prototype of an energy-efficient MAGLEV train: A step towards cleaner train transport. *Clean. Eng. Technol.* **2021**, *4*, 100217. [CrossRef]
22. Poynton, C. Frequently asked questions about color. *Retrieved June* **1997**, *19*, 2004.
23. Dorafshan, S.; Maguire, M.; Chang, M. Comparing automated image-based crack detection techniques in the spatial and frequency domains. In Proceedings of the 26th ASNT Research Symposium, Jacksonville, FL, USA, 13–16 March 2017.
24. Nigam, R.; Singh, S.K. Crack Detection in a Beam Using Wavelet Transform and Photographic Measurements. *Structures* **2020**, *25*, 436–447. [CrossRef]
25. Kumar, N. Gradient Based Techniques for the Avoidance of Oversegmentation. In Proceedings of the BEATS 2010, Jalandhar, India, 7–9 June 2010.
26. Tahir, A.; Munawar, H.S.; Akram, J.; Adil, M.; Ali, S.; Kouzani, A.Z.; Mahmud, M.A.P. Automatic Target Detection from Satellite Imagery Using Machine Learning. *Sensors* **2022**, *22*, 1147. [CrossRef] [PubMed]
27. Alipour, M.; Harris, D.K.; Miller, G.R. Robust pixel-level crack detection using deep fully convolutional neural networks. *J. Comput. Civ. Eng.* **2019**, *33*, 04019040. [CrossRef]
28. Shaukat, M.A.; Shaukat, H.; Qadir, Z.; Munawar, H.; Kouzani, A.; Mahmud, M. Cluster analysis and model comparison using smart meter data. *Sensors* **2021**, *21*, 3157. [CrossRef] [PubMed]
29. Chicco, D.; Jurman, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genom.* **2020**, *21*, 6. [CrossRef]
30. Munawar, H.S.; Hammad, A.W.A.; Haddad, A.; Soares, C.A.P.; Waller, S.T. Image-based crack detection methods: A review. *Infrastructures* **2021**, *6*, 115. [CrossRef]

31. Munawar, H.S.; Hammad, A.W.; Waller, S.T. Disaster Region Coverage Using Drones: Maximum Area Coverage and Minimum Resource Utilisation. *Drones* **2022**, *6*, 96. [CrossRef]
32. Munawar, H.S.; Mojtahedi, M.; Hammad, A.W.; Kouzani, A.; Mahmud, M.P. Disruptive technologies as a solution for disaster risk management: A review. *Sci. Total Environ.* **2022**, *806*, 151351. [CrossRef]
33. Yang, J.; Wang, W.; Lin, G.; Li, Q.; Sun, Y.; Sun, Y. Infrared thermal imaging-based crack detection using deep learning. *IEEE Access* **2019**, *7*, 182060–182077. [CrossRef]
34. Luo, Q.; Ge, B.; Tian, Q. A fast adaptive crack detection algorithm based on a double-edge extraction operator of FSM. *Constr. Build. Mater.* **2019**, *204*, 244–254. [CrossRef]
35. Özgenel, Ç.F.; Sorguç, A.G. Performance comparison of pretrained convolutional neural networks on crack detection in buildings. In Proceedings of the International Symposium on Automation and Robotics in Construction, Berlin, Germany, 20–25 July 2018.
36. Dung, C.V. Autonomous concrete crack detection using deep fully convolutional neural network. *Autom. Constr.* **2019**, *99*, 52–58. [CrossRef]
37. Moosavi, R.; Grunwald, M.; Redmer, B. Crack detection in reinforced concrete. *NDT E Int.* **2020**, *109*, 102190. [CrossRef]
38. Zou, Q.; Zhang, Z.; Li, Q.; Qi, X.; Wang, Q.; Wang, S. DeepCrack: Learning Hierarchical Convolutional Features for Crack Detection. *IEEE Trans Image Process* **2018**, *28*, 1498–1512. [CrossRef] [PubMed]
39. Chen, F.-C.; Jahanshahi, M.R. NB-CNN: Deep learning-based crack detection using convolutional neural network and Naïve Bayes data fusion. *IEEE Trans. Ind. Electron.* **2017**, *65*, 4392–4400. [CrossRef]
40. Liu, Z.; Cao, Y.; Wang, Y.; Wang, W. Computer vision-based concrete crack detection using U-net fully convolutional networks. *Autom. Constr.* **2019**, *104*, 129–139. [CrossRef]
41. Fang, F.; Li, L.; Gu, Y.; Zhu, H.; Lim, J.-H. A novel hybrid approach for crack detection. *Pattern Recognit.* **2020**, *107*, 107474. [CrossRef]
42. Zhang, A.; Wang, K.C.P.; Li, B.; Yang, E.; Dai, X.; Peng, Y.; Fei, Y.; Liu, Y.; Li, J.Q.; Chen, C. Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network. *Comput. Aided Civ. Infrastruct. Eng.* **2017**, *32*, 805–819. [CrossRef]
43. Pauly, L.; Peel, H.; Luo, S.; Hogg, D.; Fuentes, R. Deeper networks for pavement crack detection. In Proceedings of the 34th ISARC, Taipei, Taiwan, 28 June–1 July 2017.
44. Rimkus, A.; Podviezko, A.; Gribniak, V. Processing digital images for crack localization in reinforced concrete members. *Procedia Eng.* **2015**, *122*, 239–243. [CrossRef]
45. Fan, Z.; Wu, Y.; Lu, J.; Li, W. Automatic pavement crack detection based on structured prediction with the convolutional neural network. *arXiv* **2018**, arXiv:Preprint/1802.02208.
46. Lins, R.G.; Givigi, S.N. Automatic crack detection and measurement based on image analysis. *IEEE Trans. Instrum. Meas.* **2016**, *65*, 583–590. [CrossRef]
47. Munawar, H.S.; Hammad, A.W.; Waller, S.T. Remote Sensing Methods for Flood Prediction: A Review. *Sensors* **2022**, *22*, 960. [CrossRef]
48. Özgenel, Ç.F. Concrete crack images for classification. *Mendeley Data* **2018**, *1*.
49. Munawar, H.S.; Hammad, A.; Waller, S.; Thaheem, M.; Shrestha, A. An integrated approach for post-disaster flood management via the use of cutting-edge technologies and UAVs: A review. *Sustainability* **2021**, *13*, 7925. [CrossRef]
50. Talab, A.M.A.; Huang, Z.; Xi, F.; HaiMing, L. Detection crack in image using Otsu method and multiple filtering in image processing techniques. *Optik* **2016**, *127*, 1030–1033. [CrossRef]
51. Munawar, H.S.; Awan, A.A.; Khalid, U.; Maqsood, A. Revolutionizing Telemedicine by Instilling H. 265. *Int. J. Image Graph. Signal Processing* **2017**, *9*, 20–27. [CrossRef]
52. Munawar, H.S.; Maqsood, A.; Mustansar, Z. Isotropic surround suppression and Hough transform based target recognition from aerial images. *Int. J. Adv. Appl. Sci.* **2017**, *4*, 37–42. [CrossRef]
53. Akram, J.; Javed, A.; Khan, S.; Akram, A.; Munawar, H.S.; Ahmad, W. Swarm intelligence based localization in wireless sensor networks. In Proceedings of the 36th Annual ACM Symposium on Applied Computing, New York, NY, USA, 22–26 March 2021; pp. 1906–1914.
54. Ke, L.; Liu, Z.; Yu, H. Characterization of a Patch Antenna Sensor's Resonant Frequency Response in Identifying the Notch-Shaped Cracks on Metal Structure. *Sensors* **2018**, *19*, 110. [CrossRef]
55. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
56. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:Preprint/1409.1556.
57. Vedaldi, A.; Lenc, K. Matconvnet: Convolutional neural networks for matlab. In Proceedings of the 23rd ACM International Conference on Multimedia, Brisbane, Australia, 26–30 October 2015.
58. Fawcett, T. ROC graphs: Notes and practical considerations for researchers. *Mach. Learn.* **2004**, *31*, 1–38.