International Journal of Internet, Broadcasting and Communication Vol.14 No.2 47-54 (2022)
http://dx.doi.org/10.7236/IJIBC.2022.14.2.47

IJIBC 22-2-8

A study on road damage detection for safe driving of autonomous vehicles based on OpenCV and CNN

Sang-Hyun Lee

Associate Professor, Department of Computer Engineering, Honam University, Korea
leesang64@honam.ac.kr

Abstrac

For safe driving of autonomous vehicles, road damage detection is very important to lower the potential risk. In order to ensure safety while an autonomous vehicle is driving on the road, technology that can cope with various obstacles is required. Among them, technology that recognizes static obstacles such as poor road conditions as well as dynamic obstacles that may be encountered while driving, such as crosswalks, manholes, hollows, and speed bumps, is a priority. In this paper, we propose a method to extract similarity of images and find damaged road images using OpenCV image processing and CNN algorithm. To implement this, we trained a CNN model using 280 training datasheets and 70 test datasheets out of 350 image data. As a result of training, the object recognition processing speed and recognition speed of 100 images were tested, and the average processing speed was 45.9 ms, the average recognition speed was 66.78 ms, and the average object accuracy was 92%. In the future, it is expected that the driving safety of autonomous vehicles will be improved by using technology that detects road obstacles encountered while driving.

Keywords: *OpenCV, CNN, Data augmentation, Histogram Equalization, Object accuracy.*

1. Introduction

The current dissemination of autonomous vehicles is expected to take various forms that will be used as a means of transportation in the future. It is expected that self-driving cars, which have been actively used in recent years, will become a new means of transportation. In addition, it is expected to rely on autonomous vehicles rather than direct driving, and the energy of autonomous vehicles will be preferred over general autonomous vehicles because they are simple and inexpensive because electricity is used [1].

For the safe driving of autonomous vehicles, road damage detection is very important to reduce potential risks. Here, the purpose of road surface management is to improve pavement quality and prevent safety accidents [2]. As a cause of road damage, water penetrates the pavement through cracks caused by rainwater [3]. Penetrates into the compacted soil under the pavement surface and adversely affects ground subsidence. If this phenomenon continues, the quality of the pavement deteriorates and it affects the steering control of the driving vehicle, leading to a safety accident [4].

Recently, research on image processing technology based on deep learning as a sensor technology for

Manuscript Received: February. 20, 2022 / Revised: February. 22, 2022 / Accepted: February. 24, 2022
Corresponding Author: leesang64@honam.ac.kr
Tel:+82-62-940-5285, Fax: +82-62-940-5285
Associate Professor, Department of Computer Engineering, Honam University, Korea

detecting road surface conditions is being actively conducted, and it is showing high detection performance. It is an algorithm that detects road damage by applying a CNN(Convolutional Neural Network)-based calculation method to multiple layers of neural networks. This algorithm can detect pixel-level cracks and is being used as a detection technology that enables accurate road surface management [5].

Since the advent of deep learning, research on road damage detection algorithms has been conducted in the direction of improving the detection performance by improving the structure of the deep neural network. Deep neural networks have evolved by improving the combination of CNN operations to improve recognition performance. In addition, advances in GPU memory density technology have made it possible to deepen the number of neural networks, resulting in progressively improved recognition performance. However, as the neural network goes deeper, we run into the problem of loss of gradients, where the weight values no longer change. To improve this, residual networks [6] and dense networks [7] have been developed. A new neural network has been proposed to connect the inputs and outputs in a network block to update weights even in deep neural networks. This research trend is also being applied to the field of road damage detection.

A classification network that detects cracks using the residual network has been proposed [8], and a technique for dividing the crack region into pixels by applying a mixture of residual network and transfer learning has been developed [3].

And we proposed a crack detection algorithm with a new loss function considering the densely connected deep neural network and the connectivity of neighboring pixels. Many studies have been continuously conducted to improve detection performance through deep neural networks of various structures [9].

Deep learning algorithms need training data to detect road surface damage. However, unlike the learning data used in the autonomous driving field, it is not easy to obtain data when the road surface is damaged. Objects to be detected in the field of autonomous driving are objects that can be obtained under normal road conditions, such as vehicles and pedestrian traffic lights. On the other hand, road surface damage is not an object that appears on the general road surface. It is a phenomenon that occurs due to damage and aging due to long-term use, and data collection is not easy compared to other items. Just as it is not easy to protect damaged image data, it is difficult to protect label data. It is important to obtain label data for deep learning, but it takes a lot of money and time to obtain a sufficient amount of label data. In particular, in the case of a road surface, it takes a lot of time and money to secure label data in consideration of the length and area.

In this study, the OpenCV image processing technique and CNN model are used for road damage detection to be applied to autonomous vehicles.

2. Composition of learning data for road damage detection

In this chapter 2, the proposed road surface damage detection was implemented using webcam and OpenCV library. The total number of images used in this paper is 350, and the CNN model was trained using 280 training data sheets and 70 test data sheets. Figure 1 shows images of training data for road surface damage detection.

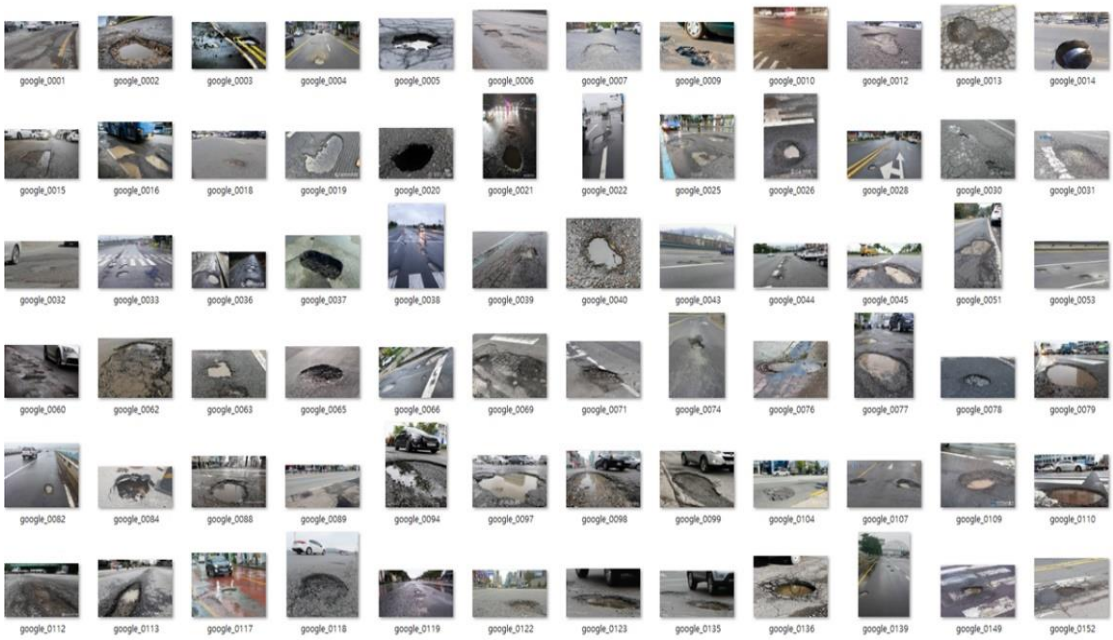

Figure 1. Example of learning data for road surface damage detection

In this paper, the input image was augmented using webcam and OpenCV library to secure insufficient training data. The data augmentation technique used has the advantage that no information loss occurs when data is augmented with the random flip technique. The data augmentation process is shown in Figure 2.

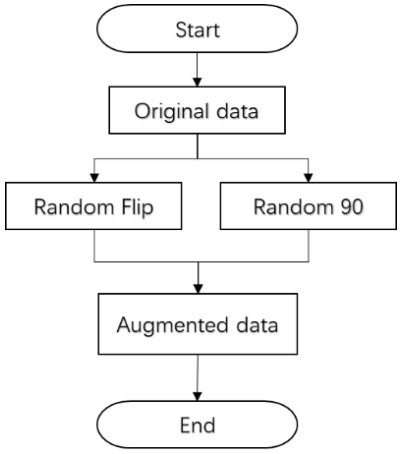

Figure 2. Data augmentation process

Histogram equalization used in data augmentation techniques generally increases the overall contrast of many images when the images are represented by a narrow range of intensity values. This adjustment better distributes the intensities across the histogram, using the entire intensity range equally. Therefore, a region with a low local contrast can obtain a higher contrast.

Table 1 is the code content to implement Histogram Equalization. Histograms are calculated from lines 1 to 4 by inputting images. Rows 6 to 12 are calculated as the minimum value of the histogram value of the image pixel.

Table 1. Image histogram code

1	def ComputeHist(img) // *define a function*
2	h, w = img.shape // *h, w = Use image.shape to check the height, width, and channel values of the image*
3	Hist, bin_edge = np.histogarm(im.reshape(1, w*h), bins=list(range(257))) // *One shape dimension can be -1. In this case, the value is inferred from the length of the array and remaining dimensions. hist() function puts a value range 257 in the form of a list.*
4	return hist
5	def computMinLevel(hist, rate, pnum): // *Declare the function computMinLevel and declare hist, rate, and pnum as variables*
6	Sum=0

7	for i in range(256):
	sum +=hist[i]
	if (sum >= (pnum*rate*0.01)):

8	return i

Therefore, we can see an example of the result of Histogram Equalization image enhancement as shown in Figure 3 with image enhancement results as in Table 1.

Figure 3. Example of comparison image before and after histogram equalization filter processing

3. Implementation and analysis of road damage detection

To train a CNN model for "detecting dynamic or static obstacles that can be encountered on the road", 280 sheets of the total data were used for training and 70 sheets were used for verification.

The implementation of the algorithm used the TensorFlow API based on Ubuntu 18.04, and the specifications of the development PC are ARM Cortex A57 Quad-Core @ 1.43GHz, LPDDR4 4GB, Maxwell architecture (128-Core, 472GFLOPS), encode 4K @ 30Hz, decode 4K @ 60Hz (H264/H265), 2x MIPI CSI-2 DPHY Lanes were used. Here, Figure 4 describes the source code for implementing road surface damage detection.

Table 2. Example of comparison image before and after histogram equaliza

1	batch_size = 32 // *Adjust the batch size to 32*
2	// *80% of the images are used for training and 20% for validation* // *Before data preprocessing, specify the batch size and width and height of the image* train_ds and val_ds = tf.keras.preprocessing.image_dataset_from_directory(data_dir, validation_split = 0.2, subset = "training", seed = 123, image_size = (img_height, img_width), batch_size = batch_size))
3	// *The AUTOTUNE option makes the tf.data runtime dynamically adjust the value at run time. The data.cache() function loads an image from disk during the first epoch and then holds it in memory, the tf.data.prefetch() function uses a background thread and an internal buffer to retrieve elements from the input dataset before the requested time.* AUTOTUNE = tf.data.experimental.AUTOTUNE train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE) val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
4	// *Here, will standardize values to be in the `[0, 1]` range by using a Rescaling layer. Keras' preprocessing utilities and tiers are subject to change.* normalization_layer = layers.experimental.preprocessing.Rescaling(1./255) data_augmentation = keras.Sequential([layers.experimental.preprocessing.RandomFlip("horizontal", input_shape=(img_height, img_width, 3)), layers.experimental.preprocessing.RandomRotation(0.05), layers.experimental.preprocessing.RandomZoom(0.05),])

5	// *Create the model the model consists of three convolution blocks with a max pool layer in each of them. There's a fully connected layer with 128 units on top of it that is activated by a `relu` activation function. This model has not been tuned for high accuracy, the goal of this tutorial is to show a standard approach.*
	num_classes = 5 # number of classes
	model = Sequential([data_augmentation, layers.experimental.preprocessing.Rescaling(1./255), layers.Conv2D(16, 3, padding='same', activation='relu'), layers.MaxPooling2D(), layers.Conv2D(32, 3, padding='same', activation='relu'), layers.MaxPooling2D(), layers.Conv2D(64, 3, padding='same', activation='relu'), layers.MaxPooling2D(), layers.Dropout(0.2), layers.Flatten(), layers.Dense(128, activation='relu'), layers.Dense(num_classes)])

6	// *Compile the model - For this tutorial, choose the `optimizers.Adam` optimizer and `losses.SparseCategoricalCrossentropy` loss function.*
	model.compile(optimizer='adam',
	loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])
	model.summary()
	epochs = 50 # Set the number of repetitions
	history = model.fit(train_ds, validation_data = val_ds, epochs = epochs) # learning

Table 2 below is the pseudocode for comparison before and after the histogram equalizer. Here, the meaning of each line as the content of the pseudo code is as follows.

1) Batch Size refers to the number of data transmitted to the network at one time among sample data. If you have 1,000 data and a batch size of 10, you will go through a total of 100 steps to create a total of 10 batch groups and 1 epoch. In general, larger batch sizes increase learning speed but increase the amount of memory.

2) It has a function to load an image from disk by using "*tf.keras.preprocessing.image_dataset_from _directory*" utility, and "*validation_split*" is recommended to be used when developing a model as a validation split option. Also, the subset is divided into training and validation.

4) Normalize the RGB channels between 0 and 1 to fit the neural network.

5) It is a function that augments data through random transformation as shown below, and serves to compensate for insufficient datasets.

6) Use a sequential model with exactly one input and one output layer in each layer. Conv 2D, a model suitable for a regular layer stack, refers to a spatial convolutional layer on an image, which is a 2D convolutional layer. MaxPooling 2D means to extract the maximum value among the values of elements entering the window. "*model.compile*" is a function that compiles a model, and "*model.summary*" is a function that outputs information.

Table 3 shows the format of the generated CNN model.

Table 3. Example of comparison

Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 180, 180, 3)	0
rescaling_1 (Rescaling)	(None, 180, 180, 3)	0
conv2d (Conv2D)	(None, 180, 180, 16)	448
max_pooling2d (MaxPooling2D)	(None, 90, 90, 16)	0
conv2d_1 (Conv2D)	(None, 90, 90, 32)	4,640
max_pooling2d_1 (MaxPooling2)	(None, 45, 45, 32)	0
conv2d_2 (Conv2D)	(None, 45, 45, 64)	18,496
max_pooling2d_2 (MaxPooling2)	(None, 22, 22, 64)	0
dropout (Dropout)	(None, 22, 22, 64)	0
flatten (Flatten)	(None, 30976)	0
dense (Dense)	(None, 128)	3,965,056
dense_1 (Dense)	(None, 5)	645

Table 4 shows a picture captured by the CNN model learning. Loss, Accuracy, val_loss, and val_accuracy shown in the contents of Figure 6 are Loss: training loss value, Accuracy: training accuracy, val_loss: test loss value, val_accuracy: test accuracy.

Table 4. Screen capture of CNN model learning

Epoch	Loss	Accuracy	Val_loss	Val_accuracy
1	2.0577	0.2249	1.5875	0.2561
2	1.5553	0.2675	1.5321	0.3171
3	1.3857	0.4590	1.4177	0.4024
4	1.2007	0.5015	1.4448	0.4390
5	1.0695	0.6049	1.3994	0.5488
6	0.9710	0.6353	1.4407	0.5244
7	0.9142	0.6383	1.3224	0.5488
8	0.8655	0.6717	1.3660	0.5244
9	0.8547	0.7052	1.4336	0.5488
10	0.7734	0.7052	1.4730	0.5488

Figure 4 shows a screen that visualizes the learned results of the proposed CNN model.

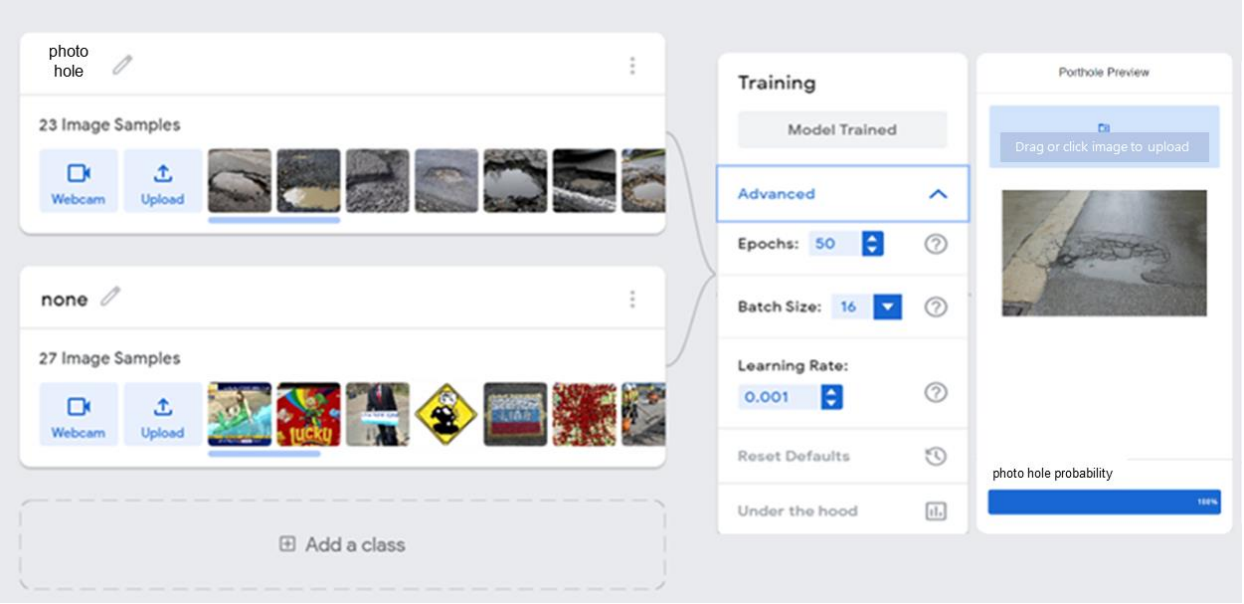

Figure 4. Visualized learning

Figure 5 shows images detected based on a model trained using a webcam and OpenCV library.

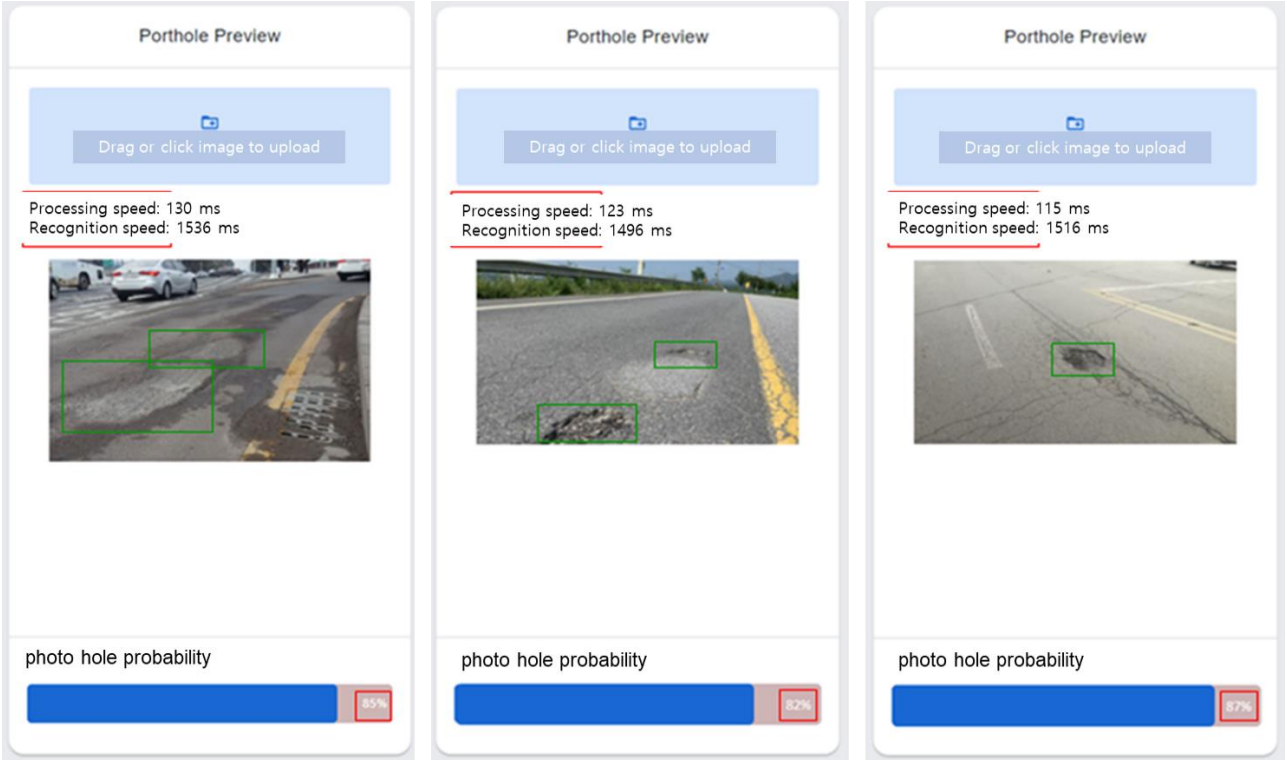

Figure 5. Images detected based on the trained model

Figure 5 shows the experimental results of object recognition processing speed and recognition speed for 100 images as the final result. As a result of the experiment, the average processing speed was 45.9ms and the average recognition speed was 66.78ms. In addition, detection was performed in 74 images out of 80 images

including portholes among the test images, and the remaining 6 images were detected incorrectly. Also, portholes were normally detected in 18 out of 20 images of general roads, and 2 were incorrectly detected. Overall, the average object accuracy is 92%.

4. Conclusion

Currently, as the number of autonomous vehicles increases, the quality control of road surfaces plays an important role in preventing safety accidents of driving vehicles. Become a factor in currently, a number of detection equipment and sensor technologies are being developed, but recently, a lot of research on road damage detection technology using image technology using deep learning is being conducted.

The part that differentiates this paper from previous studies is that OpenCV and CNN models, which can improve recognition performance even with a small number of label images, are mixed and applied to accurately detect road surface damage. In addition, it was compared with road damage recognition performance through supervised learning. As a result, as the final result, the object recognition processing speed and recognition speed of 100 images were tested. The average processing speed was 45.9ms and the average recognition speed was 66.78ms, and the average object accuracy was 92%. have. If such OpenCV and CNN models are mixed and applied to an embedded system, it is expected that in the future, it will be installed in autonomous vehicles and can play an important role in preventing safety accidents by more accurately and efficiently managing road surface damage areas in real time.

References

[1] M. Tinnila and J. Kalli, "Impact of future trends on personal mobility services," *International Journal of Automotive Technology and Management*, vol. 15, no. 4, pp.401-417, 2015.
DOI: https://doi/pdf/10.1504/IJATM.2015.072876

[2] R, Fan, "Real-time computer stereo vision for automotive applications," Doctoral Dissertation, University of Bristol, England, 2018.

[3] S. Bang, S. Park, H. Kim, "Encoder-decoder network for pixel level road crack detection in black-box images," *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 8, pp.713-727, 2019.
DOI: https://doi.org/10.1111/mice.12440

[4] R. Madli, S. Hebbar, P. Pattar and V. Golla, "Automatic detection and notification of potholes and humps on roads to aid drivers," *IEEE Sensors Journal*, vol. 15, no. 8, pp.4313-4318, 2015.
DOI: https://doi.org/10.1109/JSEN.2015.2417579.

[5] M. D. Jenkins, T. A. Carr, M. I. Iglesias, T. Buggy and G. Morison, "A deep convolutional neural network for semantic pixel-wise segmentation of road and pavement surface cracks," In Proc. *26th European Signal Processing Conference (EUSIPCO*), Rome, Italy, pp.2120-2124, 2018.
DOI: https://doi.org/10.23919/EUSIPCO.2018.8553280

[6] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," In Proc. *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, pp.770-778, 2016.
DOI: https://doi.org/*arXiv:1512.03384*

[7] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely connected convolutional networks," In Proc. *IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, pp.4700-4708, 2017.
DOI: https://doi.org/arXiv:1608.06993

[8] H. Feng, G. S. Xu and Y. Guo, "Multi-scale classification network for road crack detection," *IET Intelligent Transport Systems*, vol. 13, no. 2, pp.398-405, 2018.
DOI: https://doi.org/10.1049/iet-its.2018.5280

[9] Q. Mei, M. Gul and M. R. Azim, "Densely connected deep neural network considering connectivity of pixels for automatic crack detection," *Automation in Construction*, vol. 110, p.103018, 2020.
DOI: https://doi.org/10.1016/j.autcon.2019.103018