Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 207 (2022) 3948–3956

Procedia Computer Science

www.elsevier.com/locate/procedia

26th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2022)

A Lightweight CNN-Based Vision System for Concrete Crack Detection on a Low-Power Embedded Microcontroller Platform

Laura Falaschetti[a,*], Mattia Beccerica[b], Giorgio Biagetti[a], Paolo Crippa[a], Michele Alessandrini[a], Claudio Turchetti[a]

[a]*DII - Department of Information Engineering, Università Politecnica delle Marche, via Brecce Bianche, 12, I-60131 Ancona, Italy*

[b]*Università Politecnica delle Marche, via Brecce Bianche, 12, I-60131 Ancona, Italy*

Abstract

The detection of cracks is a key aspect for assessing the condition of in-service structures such as road, bridges or dams. Therefore it assumes a great importance for civil infrastructures monitoring, road maintenance and traffic safety. Intelligent detection methods based on convolutional neural networks (CNNs) have been widely applied to the field of crack detection in recently years. In this work we propose a vision system based on two lightweight and accurate CNN models implemented in a low-cost, low-power platform, namely the OpenMV Cam H7 Plus, to monitor and to detect concrete cracks in real-time, suitable to realize a prototype of early warning system. In order to be useful, such a system must provide a very high accuracy, so as not to give false alarms, and be parsimonious enough on computational resources to be embedded into low-power, portable systems that can be deployed on the field. To reach this goal, firstly we analyze different state-of-the-art CNNs applied to the concrete crack detection task in order to discover the smallest network in terms of memory storage and number of parameters. Then, we compare the performance, in terms of memory occupancy and accuracy, of the proposed CNN architectures with the smallest network in the investigated literature, LeNet, all trained on two different image datasets, the Concrete Crack Images for Classification dataset and the SDNET2018 dataset, and implemented on the embedded system OpenMV Cam H7 Plus. The proposed CNN architectures perform nicely on this platform, using only a small fraction, between 6% to 26%, of the memory required by LeNet, and always providing better accuracy in all the tested cases and on both the datasets tried, with only a marginal increase in inference time.

Keywords: Convolutional neural network (CNN); deep learning (DL); crack detection; vision system; classification; microcontroller; embedded system.

* Corresponding author. Tel.: +39-071-220-4129 ; fax: +39-071-220-4464.
E-mail address: l.falaschetti@univpm.it

1877-0509
10.1016/j.procs.2022.09.457

1. Introduction

Civil infrastructures when are progressively approaching their life span expectancy need to be checked against the integrity of their structural system. Crack is one of the most important and the most easy to evaluate indicator for assessing the condition of in-service structures such as road, bridges or dams [1].

Manual detection of cracks in concrete structures or road pavements consists of subjective visual inspections and evaluations by human experts, and it is a time-consuming, expensive, and inspector-dependent activity. Automatic detection techniques based on computer vision algorithms are an optimal solution, which eliminates subjectivity and reduces processing times and costs [2, 3].

Recently, convolutional neural networks (CNN) have been implemented instead of computer vision algorithms to recognize and classify images [4], CNNs are composed of deep layers of operations, in which the filters of the layers learn features automatically and hierarchically from raw data, and therefore they are often referred to as deep learning (DL). DL systems benefit greatly from extremely fast computations, so implementations on graphics processing units (GPUs) [5–8], as well as cloud-based computing services [9] or fast CPUs [10] have become standardized.

Nevertheless, GPUs or fast CPUs are quite expensive and power hungry, cloud access might not always be available and is still energy expensive, therefore low-power microcontroller implementation of the above networks are extremely interesting to be investigated. Thus, concrete crack detection models based on CNNs have been studied, and efficient automatic intelligent systems able to work on microcontrollers have been developed and implemented.

More in the detail, the aim of this paper is to propose a vision system based on two lightweight and accurate CNN models implemented in a low-cost, low-power platform, namely the OpenMV Cam H7 Plus, to monitor and to detect concrete cracks in real-time, thus suitable to realize a prototype of early warning system. To reach this goal we proceeded as follows. Several state-of-the-art CNNs are preliminarily analysed and a comparative study has been carried out to determine the method with the best performance, both in terms of memory cost and parameter numbers (model complexity), without loss in accuracy. The resulting best CNN has been used as a benchmark to evaluate the performance of the proposed method. Two lightweight CNN architectures are derived and trained using the Concrete Crack Images for Classification dataset and the SDNET2018 dataset, both containing images of different surfaces with or without cracks. The proposed architectures are implemented in the Python programmable OpenMV Cam H7 Plus platform, to meet the requirements of a low-power, low-cost, real-time image sensor for the concrete crack detection task. Moreover, a comparison between the proposed CNNs and the benchmark model in terms of model complexity has been conducted, both on desktop and on the embedded platform, to validate the proposed approach.

The article is organized as follows. In Section 2 we describe the image datasets we used, the CNN architectures and the hardware and software used. Section 3 reports the experimental results on both the desktop and the embedded platform OpenMV Cam H7 Plus. Finally some conclusions are drawn in Section 4.

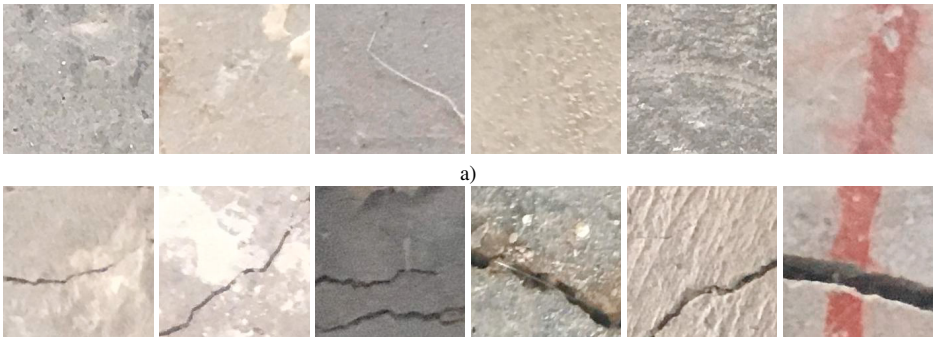
2. Material and Method

2.1. Datasets

Two datasets of concrete crack images have been used in our investigation.

The first, the Concrete Crack Images for Classification dataset [11, 12] ("Concrete Crack" in short) contains images of concrete surfaces possibly having cracks. The data was collected from various campus buildings at the Middle East Technical University (METU) in Ankara, Turkey, and is divided into two sets, negative and positive, i.e., without or with cracks, respectively, for image classification purposes. Each class contains 20000 images for a total of 40000 RGB images, each 227×227 pixels, extracted from 458 high-resolution images (4032×3024 pixels) with the method [13]. The images contained in this dataset depict a variety of different surface finishes, as well as varied illumination conditions as shown in Fig. 1.

The second is SDNET2018 [14], an annotated dataset of concrete images collected from Utah State University. It consists of over 56000 color images containing both cracked and non-cracked pictures from three types of structures: pavements, walls, and bridge decks. It includes cracks with a variety of sizes ranging from 0.06 mm to 25 mm. The images in the dataset may include many different kinds of obstructions, such as shadows, stains, different surface



a)

b)

Fig. 1. Example of images from the "Concrete Crack" dataset, belonging to the two different classes: a) negative, b) positive.

roughness, joints and scaling, edges, holes, and background debris as shown in Fig. 2. Each image is 256 × 256 pixels in size, and is annotated as "with crack" or "without crack" for the supervised training.

a) SDNET2018 Pavement

b) SDNET2018 Wall

c) SDNET2018 Bridge Deck

Fig. 2. Example of images from the "SDNET2018" dataset that contains three subsets with non-cracked and cracked pictures from three types of structures: a) pavement, b) wall, c) bridge deck.

2.2. CNN Architecture

The proposed CNN architecture V1 is depicted in Fig. 3 and comprises 12 layers with 7 weight layers in total. Specifically, it consists of 4 convolutional layers each followed by a ReLU activation function and two of them followed by a max-pooling operation, plus 3 fully-connected layers with a final softmax classifier. In the final stage two dropout layers, with a dropout rate set at 0.5, have been introduced to reduce the overfitting, which occurs when the network tends to specialize too much on the training set, thus resulting in worse classification performance. A detailed description of this architecture is provided in Table 1.

The proposed CNN architecture V2 is depicted in Fig. 4 and comprises 13 layers with 6 weight layers in total. Specifically, it consists of 4 convolutional layers each followed by a ReLU activation function and two of them followed by a max-pooling operation, plus 2 fully-connected layers with a final softmax classifier. In the final stage four dropout layers, with a dropout rate set in the range 0.2 - 0.5, have been introduced. With respect to the CNN V1, the introduction of the two additional dropout layers considerably reduces the number of parameters. A detailed description of this architecture is provided in Table 2.

Table 1. CNN V1 model summary.

Layer type	Output shape	Param
conv_1	$62 \times 62 \times 16$	160
relu_1	$62 \times 62 \times 16$	0
conv_2	$60 \times 60 \times 16$	2320
relu_2	$60 \times 60 \times 16$	0
maxpool_1	$20 \times 20 \times 16$	0
conv_3	$18 \times 18 \times 32$	4640
relu_3	$18 \times 18 \times 32$	0
conv_4	$16 \times 16 \times 32$	9248
relu_4	$16 \times 16 \times 32$	0
maxpool_2	$8 \times 8 \times 32$	0
flatten	None $\times$ 2048	0
dropout_1	None $\times$ 2048	0
dense_1	None $\times$ 32	65568
relu_5	None $\times$ 32	0
dropout_2	None $\times$ 32	0
dense_2	None $\times$ 32	1056
relu_6	None $\times$ 32	0
dense_3	None $\times$ 2	66
softmax	None $\times$ 2	0

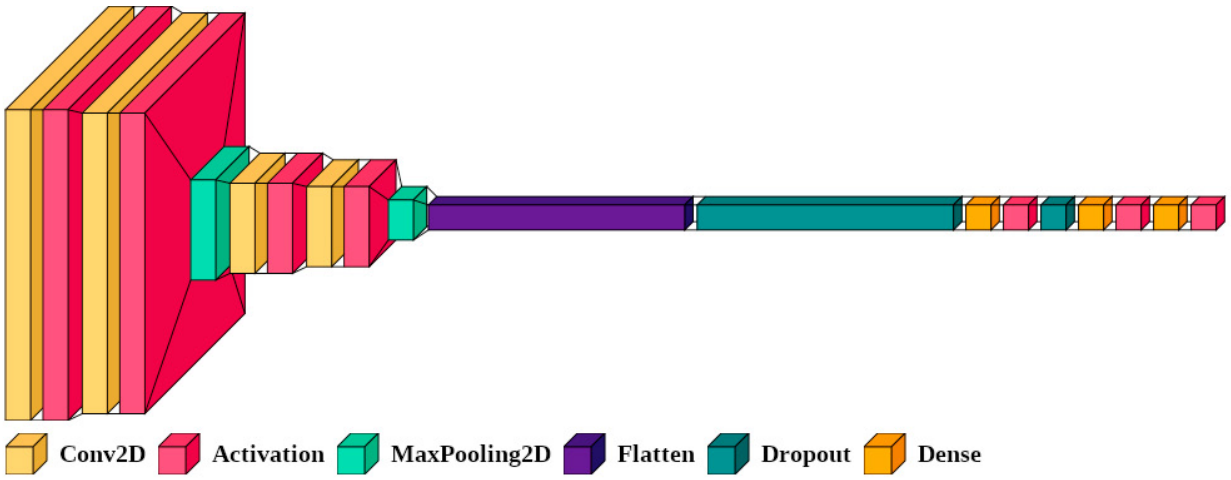


Fig. 3. CNN V1 architecture.

The networks were trained for 20 epochs on the previously described “Concrete Crack” and “SDNET2018” (pavement, wall, bridge deck) dataset with images of size 64×64, by using an Adadelta optimizer with categorical cross entropy and a batch size of 64.

Table 2. CNN V2 model summary.

Layer type	Output shape	Param
conv_1	$62 \times 62 \times 16$	160
relu_1	$62 \times 62 \times 16$	0
conv_2	$60 \times 60 \times 16$	2320
relu_2	$60 \times 60 \times 16$	0
maxpool_1	$20 \times 20 \times 16$	0
dropout_1	$20 \times 20 \times 16$	0
conv_3	$18 \times 18 \times 16$	2320
relu_3	$18 \times 18 \times 16$	0
conv_4	$16 \times 16 \times 16$	2320
relu_4	$16 \times 16 \times 16$	0
maxpool_2	$5 \times 5 \times 16$	0
dropout_2	$5 \times 5 \times 16$	0
flatten	None $\times 400$	0
dropout_3	None $\times 400$	0
dense_1	None $\times 16$	6416
relu_7	None $\times 16$	0
dropout_4	None $\times 16$	0
dense_3	None $\times 2$	34
softmax	None $\times 2$	0

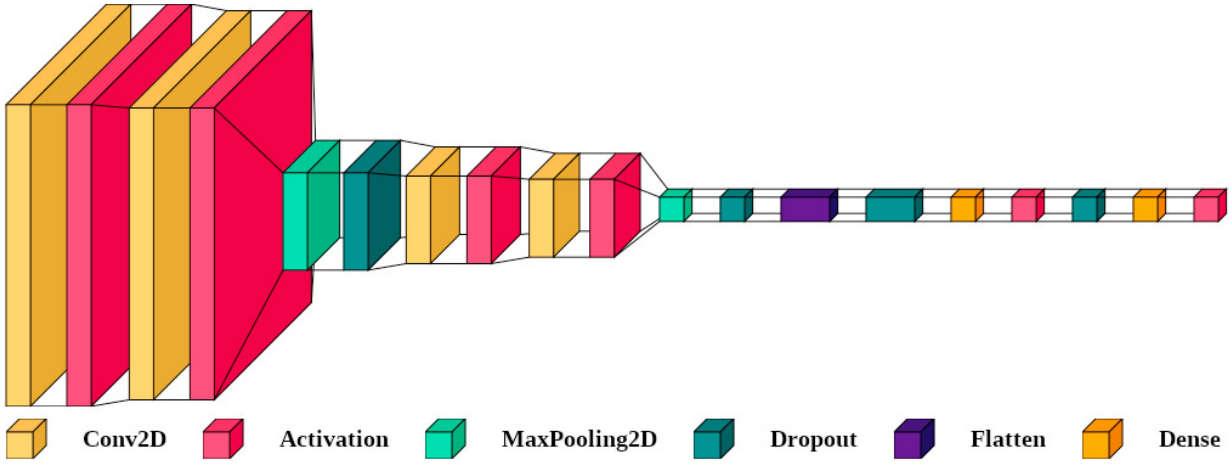


Fig. 4. CNN V2 architecture.

2.3. Hardware and Software

The OpenMV Cam STM32H7 Plus camera[1] is a low-cost, low-power (0.8 W) Python programmable machine vision camera that supports an extensive set of image processing functions and neural networks. It is based on the

[1] https://openmv.io/products/openmv-cam-h7-plus

STM32H743II ARM Cortex-M7 MCU, running at 480 MHz and featuring over 1 MiB of on-chip SRAM, 2 MB of on-chip FLASH, complemented with 32 MiB of off-chip SDRAM and other 32 MB of off-chip FLASH. It is also equipped with an OV5640 image sensor that can capture images up to size 2592 × 1944, though most algorithms run between 10-15-25-50 FPS on a QVGA (320 × 240) resolution or less.

The experiments were conducted using the TensorFlow v. 2.5.0 and Keras v. 2.5.0 to train the models, using Google Colaboratory, with the "Concrete Crack" and the "SDNET2018" datasets partitioned as reported in Table 3, and tested on the board with the OpenMV firmware v. 4.1.1.

The OpenMV Cam is particularly suitable for the implementation of machine learning applications since it can be directly programmed in high level Python scripts, thanks to the MicroPython Operating System. There is a small internal filesystem on the OpenMV Cam which is stored within the microcontroller's flash memory. Therefore, the model can be directly loaded and run by the OpenMV H7 Plus Cam, with the constraint that the model must fit within the available 31 MB in the frame buffer RAM. The supported model is a quantized TensorFlow Lite for microcontrollers model (tflite file format), thus, once the float32 Keras model (h5 file format) has been trained to a satisfactory accuracy, it must be converted to an int8 tflite model to run on the embedded device. Fig. 5 describes the process of integrating the CNN into the OpenMV environment.

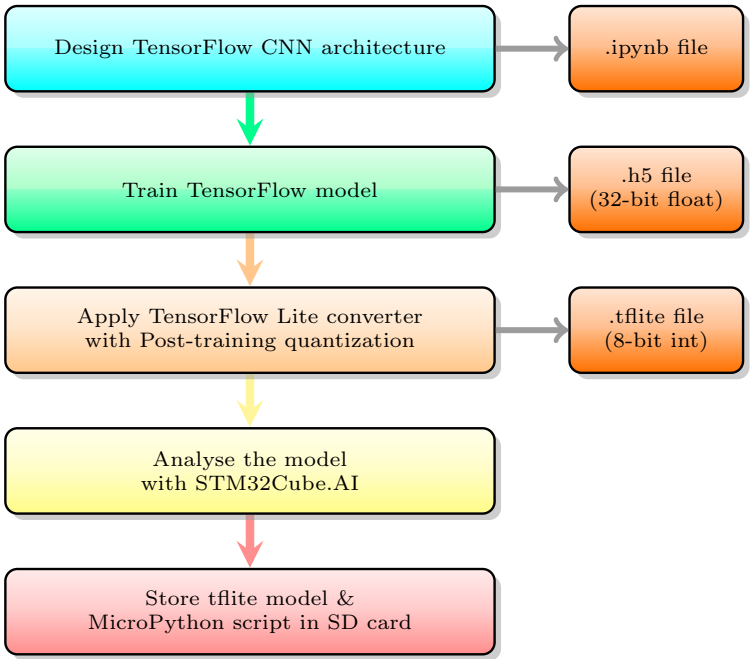


Fig. 5. OpenMV Cam code flow.

3. Experimental Results

In order to show the validity of the described models, firstly the performance achieved on PC has been considered and then the models have been evaluated on the chosen embedded system, the OpenMV Cam H7 Plus. The first experiment aims to compare the state-of-the art CNNs in order to find the best one in terms of memory occupancy and then we compare the performance of the implemented CNNs (V1, V2) with the chosen network. The second aims to determine the final performance, obtained by the network implemented on the embedded system. The experiments were conducted using the TensorFlow v. 2.5.0 and Keras v. 2.5.0 to train the models on Google Colaboratory (GPU runtime) with the "Concrete Crack" and the "SDNET2018" datasets partitioned as reported in Table 3 and the OpenMV firmware v. 4.1.1 on board.

Table 3. Consistency of the datasets and of the partitions considered in the experimentation, for training, validation and testing.

Dataset	Training Samples (70%)		Validation Samples (15%)		Testing Samples (15%)		Total Samples
	cracked	not cracked	cracked	not cracked	cracked	not cracked	
Concrete Crack	14000	14000	3000	3000	3000	3000	40000
SDNET2018 Pavement	1826	15208	391	3259	391	3259	24334
SDNET2018 Wall	2696	10001	578	2143	577	2143	18138
SDNET2018 Bridge Deck	1418	8116	304	1739	303	1739	13619

3.1. Test on desktop

In this experiment several state-of-the-art CNN architectures have considered in order to compare their performance with those achieved with the proposed CNNs (V1, V2). In particular the following networks have been used: ResNet [15], MobileNet V1 [16] , MobileNet V2 [17], MobileNet V2 [17], PeleeNet [18], SqueezeNet [19] and LeNet [20].

Firstly we conducted an investigation on the best CNNs in terms of memory occupancy, since our goal is to implement a CNN on a low-power, low-cost, memory constrained microcontroller. Table 4 reports a comparison of the state-of-the-art CNNs trained on the "Concrete Crack" dataset, showing that the LeNet architecture reaches the smaller memory footprint, maintaining a good accuracy. Thus, we chose this architecture for the performance comparison with the proposed CNNs (V1, V2).

Table 4. Comparison of the state-of-the-art CNNs trained on the "Concrete Crack" dataset — Performance on desktop — h5 model.

Model	Memory [MB]	Parameters	Accuracy [%]
ResNet [15]	90.323	23581186	99.45
MobileNet V1 [16]	21.755	3237058	**99.87**
MobileNet V2 [17]	16.201	2268226	**99.87**
PeleeNet [18]	26.174	2113250	98.23
SqueezeNet [19]	5.843	736450	99.08
LeNet [20]	**3.918**	**337806**	98.95

Table 5 shows the results achieved comparing the proposed networks with LeNet, in terms of memory cost, number of parameters and test accuracy. As you can see, the proposed networks outperform LeNet architecture both for memory storage, computational complexity and accuracy

3.2. Test on the embedded platform OpenMV Cam H7 Plus

In order to validate that the proposed CNNs are suitable to be implemented in a low-cost embedded microcontroller, we evaluated the final CNNs performance obtained by the networks implemented in the embedded system OpenMV Cam H7 Plus.

Table 6 reports the results achieved for memory cost, number of parameters, accuracy, inference time and number of recognized frames per second (FPS). All the networks are compressed by a factor of 4 due to the post training quantization that is performed by the TensorFlow Lite converter Python API. Also in this case, the proposed CNNs outperform the LeNet architecture in terms of memory cost, number of parameters and accuracy. As far as the inference time and FPS is concerned, the values achieved with the proposed CNNs are just a few FPS below the LeNet performance. This does not constitute a real issue since the crack detection task does not require a large number of

Table 5. Comparison of the proposed CNNs to the most compact network from the analyzed literature, LeNet, on the "Concrete Crack" dataset and the "SDNET2018" dataset (pavement, wall, bridge deck) — Performance on desktop — h5 model.

Model	**Memory [MB]**	**Parameters**	**Accuracy [%]**			
			Concrete Crack	SDNET2018 Pavement	SDNET2018 Wall	SDNET2018 Bridge Deck
Proposed CNN V1	1.027	83058	**99.50**	90.00	**83.49**	**84.62**
Proposed CNN V2	**0.222**	**13570**	99.33	**90.03**	80.12	82.86
LeNet	3.918	337806	98.95	86.74	79.01	83.45

recognized frames per second. Moreover, the first aim of this work is to obtain a lightweight CNN to be implemented in a resource constrained embedded system, like the OpenMV Cam H7 Plus.

Table 6. Comparison of the proposed CNNs to the most compact network from the analyzed literature, LeNet, on the "Concrete Crack" dataset and the "SDNET2018" dataset (pavement, wall, bridge deck) — Performance on OpenMV Cam H7 Plus — tflite model.

Model	**Memory [MB]**	**Parameters**	**Accuracy [%]**				**Inference time [ms]**	**FPS [s^{-1}]**
			Concrete Crack	SDNET2018 Pavement	SDNET2018 Wall	SDNET2018 Bridge Deck		
Proposed CNN V1	0.087	83058	**99.47**	**90.00**	**80.29**	**84.23**	90.28	11.07
Proposed CNN V2	**0.019**	**13570**	99.33	89.34	78.71	82.47	74.43	13.43
LeNet	0.327	337806	98.90	86.30	78.97	83.15	**61.21**	**16.33**

4. Conclusion

In this work we analyze different neural networks-based systems to detect the presence of cracks in concrete slabs or surfaces. The methods are based on image recognition, so that they can be applied to automated, possibly continuous, structural integrity monitoring, and provide early warning as soon as cracks are detected. In order to be useful, such a system must provide a very high accuracy, so as not to give false alarms, and be parsimonious enough on computational resources to be embedded into low-power, portable systems that can be deployed on the field.

The proposed CNN architectures fill nicely this spot, using only between 6% to 26% of the memory required by the smallest of the network found in the literature, LeNet, and always providing better accuracy in all the tested cases and on both the datasets tried, with only a marginal increase in inference time.

Future works regard the improvement of the achieved accuracy and the application of other compression techniques, in addition to the used tflite quantization, such as pruning and/or tensor decomposition techniques. The aim is to improve the CNN architectures achieving a higher accuracy and then apply these compression techniques to the implemented CNNs in order to further reduce the memory occupancy.

References

[1] W. Xi, J. Yang, J. Song, X. Qu, Deep learning model of concrete dam deformation prediction based on CNN, IOP Conference Series: Earth and Environmental Science 580 (1) (2020) 012042.

[2] Y. Shi, L. Cui, Z. Qi, F. Meng, Z. Chen, Automatic road crack detection using random structured forests, IEEE Transactions on Intelligent Transportation Systems 17 (12) (2016) 3434–3445.

[3] W. Cao, Q. Liu, Z. He, Review of pavement defect detection methods, IEEE Access 8 (2020) 14531–14544.

[4] L. Falaschetti, L. Manoni, R. C. F. Rivera, D. Pau, G. Romanazzi, O. Silvestroni, V. Tomaselli, C. Turchetti, A low-cost, low-power and real-time image detector for grape leaf esca disease based on a compressed cnn, IEEE Journal on Emerging and Selected Topics in Circuits and Systems 11 (3) (2021) 468–481.

[5] L. Ali, F. Alnajjar, H. A. Jassmi, M. Gocho, W. Khan, M. A. Serhani, Performance evaluation of deep CNN-based crack detection and localization techniques for concrete structures, Sensors 21 (5).

[6] L. Deng, H.-H. Chu, P. Shi, W. Wang, X. Kong, Region-based CNN method with deformable modules for visually classifying concrete cracks, Applied Sciences (Switzerland) 10 (7).

[7] W. Choi, Y.-J. Cha, SDDNet: Real-time crack segmentation, IEEE Transactions on Industrial Electronics 67 (9) (2020) 8016–8025.

[8] S. Dorafshan, R. J. Thomas, M. Maguire, Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete, Construction and Building Materials 186 (2018) 1031–1045.

[9] M. A. Mohammed, Z. Han, Y. Li, Exploring the detection accuracy of concrete cracks using various CNN models, Advances in Materials Science and Engineering 2021.

[10] K. Hacefendiolu, H. B. BaÅaa, Concrete road crack detection using deep learning-based faster R-CNN method, Iranian Journal of Science and Technology - Transactions of Civil Engineering 46 (2) (2022) 1621 1633.

[11] Ç. F. Özgenel, Concrete crack images for classification, http://dx.doi.org/10.17632/5y9wdsg2zt.2, mendeley Data, V2, Accessed: 2022-05-01 (2019).

[12] Ç. F. Özgenel, A. G. Sorguç, Performance comparison of pretrained convolutional neural networks on crack detection in buildings, in: Proceedings of the 35th International Symposium on Automation and Robotics in Construction (ISARC), International Association for Automation and Robotics in Construction (IAARC), 2018, pp. 693–700.

[13] L. Zhang, F. Yang, Y. Daniel Zhang, Y. J. Zhu, Road crack detection using deep convolutional neural network, in: 2016 IEEE International Conference on Image Processing (ICIP), 2016, pp. 3708–3712.

[14] S. Dorafshan, R. J. Thomas, M. Maguire, SDNET2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks, Data in Brief 21 (2018) 1664–1668.

[15] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, CoRR abs/1512.03385.

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, CoRR abs/1704.04861.

[17] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, L. Chen, Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation, CoRR abs/1801.04381.

[18] R. J. Wang, X. Li, S. Ao, C. X. Ling, Pelee: A real-time object detection system on mobile devices, CoRR abs/1804.06882.

[19] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, K. Keutzer, Squeezenet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, CoRR abs/1602.07360.

[20] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.