Automation in Construction 125 (2021) 103606

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Automatic crack classification and segmentation on masonry surfaces using convolutional neural networks and transfer learning

Dimitris Dais [a,b,*], İhsan Engin Bal [a], Eleni Smyrou [a], Vasilis Sarhosis [b]

[a] Research Center for Built Environment NoorderRuimte, Hanze University of Applied Sciences, Zernikeplein 11, 9701 DA Groningen, the Netherlands
[b] School of Civil Engineering, University of Leeds, Woodhouse, LS2 9JT Leeds, UK

ARTICLE INFO

Keywords:
CNN
Masonry
Crack detection
Segmentation
Classification
Transfer learning
Deep learning

ABSTRACT

Masonry structures represent the highest proportion of building stock worldwide. Currently, the structural condition of such structures is predominantly manually inspected which is a laborious, costly and subjective process. With developments in computer vision, there is an opportunity to use digital images to automate the visual inspection process. The aim of this study is to examine deep learning techniques for crack detection on images from masonry walls. A dataset with photos from masonry structures is produced containing complex backgrounds and various crack types and sizes. Different deep learning networks are considered and by leveraging the effect of transfer learning crack detection on masonry surfaces is performed on patch level with 95.3% accuracy and on pixel level with 79.6% F1 score. This is the first implementation of deep learning for pixel-level crack segmentation on masonry surfaces. Codes, data and networks relevant to the herein study are available in: github.com/dimitrisdais/crack_detection_CNN_masonry.

1. Introduction

Brick masonry is one of the main structural components in modern and historical structures along the world. Numerous old masonry buildings still exist proving that when well-preserved, the life cycle of such structures may be significantly extended [1]. In many cases, historical masonry structures have been found to be vulnerable to seismic excitations and thus thorough damage assessment is required to propose suitable restoration schemes, when necessary [2]. Moreover, masonry has been widely used in modern structures in countries with low or no seismicity. When masonry structures subjected to induced seismicity, like the ones in the north of The Netherlands, have been found to be susceptible to seismic excitations given the fact that they were constructed without any seismic design [3]. As another example of vulnerable masonry structural systems, arch bridges are the most common single bridge type on the UK rail network, most of which are now over than 100 years old and showing significant signs of distress. The importance to develop improved analysis and assessment methods for these bridges was highlighted [4]. In brief, masonry structures need to be properly inspected to detect any defects on early stage or after an extreme event in order to safeguard them.

Manual inspection is the most common practice due to its simplicity and the lack of reliable alternatives. Nevertheless, this practice is rather laborious, slow and expensive when accounting for the man-hours required to be invested in the field and at the office to process the obtained data. On top of that, the quality of the process can be subjective since it heavily relies on the skills and the physical condition of the inspector as well as lack of experience or tiredness could easily lead to ill-reported damage. Manual inspection can raise safety concerns since there are parts of the structures with access restrictions and difficult to reach. The manual inspection becomes particularly difficult for the post-event cases, such as in the catastrophic aftermath of a strong earthquake, when a high number of buildings need to be inspected with limited resources in a short time. Apart from the efficiency, reliability is another aspect to be considered when inspecting masonry structures manually. Significant variability in the routine inspection documentation of structural conditions was previously reported [5,6]. Discrepancy was observed both for the assignment of condition ratings but also for the prepared documents, e.g. field inspection notes, photographs, etc. Specifically, on average between four and five different condition rating values were assigned to each structural component, with a maximum of six being assigned [5].

* Corresponding author at: Research Center for Built Environment NoorderRuimte, Hanze University of Applied Sciences, Zernikeplein 11, 9701 DA Groningen, the Netherlands.
E-mail address: d.dais@pl.hanze.nl (D. Dais).

https://doi.org/10.1016/j.autcon.2021.103606
Received 27 July 2020; Received in revised form 26 January 2021; Accepted 28 January 2021
Available online 27 February 2021

In order to address the drawbacks of manual inspection, vision-based assessment and monitoring of civil infrastructures are gaining ground [7]. In particular, computer vision for crack detection has interested researchers for quite some time. Vision-based crack detection is a perfect example of non-destructive assessment technique, which can be useful especially for historical structures where strict regulations apply and even simple interventions, such as placing crack-rulers, are not permitted by the conservation authorities.

Deep Learning (DL), which is a subfield of artificial intelligence, and its representative tool, namely Convolutional Neural Network (CNN) have proven their efficacy in object detection [8]. Unlike traditional machine learning approaches, DL does not require any hand-crafted features and thus provides end-to-end classifiers which internally learn features and can automatically detect objects [9]. This attribute of DL algorithms along with the recent development of the graphics processing units (GPU) which allowed for very fast computations have boosted their usage in different fields. For the case of crack detection from images, the user only provides as input different photos and receives as output any detected cracks in these photos without the necessity for any manual intervention. DL for crack detection has found different case studies such as on inspection of bridges [10], gas turbines [11] and asphalt surfaces [12].

The aim of this paper is to examine different DL techniques for crack detection on images from masonry walls. Recent developments in DL for crack detection and successful techniques are highlighted in Section 2.1 while studies for vision-based assessment on masonry surfaces found in the literature are presented in Section 2.2. In order to address the lack of data in the literature, a dataset with photos from masonry structures is produced containing complex backgrounds and various crack types and sizes (Section 3). Since for masonry structures little work has been done for crack detection it is deemed beneficial to train networks both for patch classification (Section 4) and pixel-level segmentation (Section 5) in order to examine the efficacy of different techniques and broadcast the feasibility of DL methods on crack detection for masonry surfaces. To the authors' best knowledge, this study is the first implementation of DL for pixel-level crack segmentation on masonry surfaces. The technique of transfer learning is also leveraged in order to improve the performance of the DL networks for crack detection on patch and pixel level. Finally, a comparative study is performed where a segmentation network trained on masonry images is tested on photos with cracks taken from concrete surfaces in order to evaluate the ability of CNNs to generalize over different materials (section 6). Codes, data and networks relevant to the herein study can be found in the GitHub repository: github.com/dimitrisdais/crack_detection_CNN_masonry.

2. Related work

2.1. Convolutional neural networks for crack detection

Image classification with CNN can be categorized into three types: image patch classification, boundary box regression and semantic segmentation [13]. In image patch classification the image is divided in patches and each patch is labelled with a class (Fig. 1a). When boundary box regression is considered, a box bounds the detected object, that is a crack, and reveals its position and boundaries (Fig. 1b). These two classification techniques have been extensively used to detect cracks and other defects, and have shown promising results [11,14–16]. Nevertheless, these techniques are implemented at block level rather than at pixel level. On the contrary, semantic segmentation provides information about the exact location, width or length of any defects/cracks since each pixel is assigned to a class label (Fig. 1c) [17–20]. Pixel-wise image segmentation has gained ground in the recent years over image patch classification and boundary box regression. A review on DL methods for semantic segmentation applied to various application areas was presented in [21].

Recently Fully Convolutional Networks (FCNs), which are end-to-end networks, have been extensively used for semantic segmentation [22]. FCNs performed as an extended CNN where the final prediction was an image with semantic segmentation instead of a class identification. In a recent study, FCNs have been implemented for semantic segmentation on concrete crack images by evaluating several pre-trained network architectures serving as the backbone of FCN encoder [23]. FCNs were also used by Yang et al. [24] to semantically identify and segment crack pixels with different scales and were combined with morphological operations to extract geometric characteristics, such as length and width, directly from images without manual measurement. An automatic crack segmentation method based on CNN and consisting of the extended FCN and the Deeply-Supervised Nets (DSN) was introduced by Liu et al. [17]. Special care was given to produce a dataset of photos from asphalt and concrete surfaces with cracks in multi-scale and multi-scene to evaluate the crack detection systems. A modified FCN with fine-tuning the DenseNet-121 (a densely connected CNN) was implemented by Li et al. [25] to provide pixel-level detection of multiple damages, i.e. crack, spalling, efflorescence and holes, found on concrete surfaces. The suggested method outperformed the results obtained from a method based on SegNet (a deep convolutional encoder–decoder architecture trained to classify urban street pictures at pixel level) while producing smaller sizes of trained models as well. An FCN was implemented to simultaneously identify material type (concrete, steel, asphalt), as well as fine (cracks, exposed rebar) and coarse (spalling, corrosion) structural damage [26].

U-net is a deep FCN that was developed for biomedical image segmentation and outperformed other state of the art networks [27]. Since then, U-net has become a benchmark for image segmentation in different fields and its efficacy to detect thin edges resulted in its wide implementation on the inspection of structures. In particular, U-net was implemented for crack detection on pavement surfaces [28,29]. Another showcase for U-net was presented by Liu et al. [30] for concrete crack detection, where the U-net performed better than the other FCN methods [23,24] while being trained on significantly smaller training

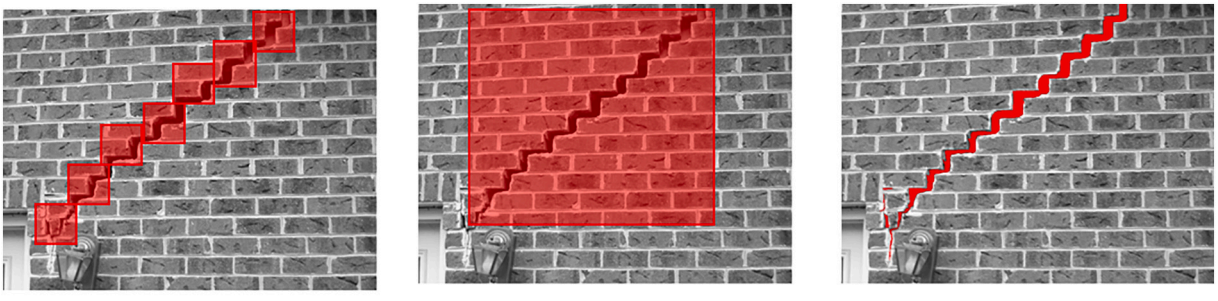

Fig. 1. Crack detection with (a) image patch classification, (b) boundary box regression and (c) semantic segmentation. The output of each crack detection technique is denoted with red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

sets. Note that in order to solve the sample imbalance problem, the focal loss function was selected in the study by Lui et al. [30]. A DL algorithm based on U-Net and a CNN with alternately updated clique (CliqueNet), called U-CliqueNet, was proposed to separate cracks from background in tunnel images [31]. The system obtained promising results and was able to separate cracks from images with noise similar to cracks, such as patchwork joints, wires, etc. It is noted that, while other studies were based on datasets of couple of hundreds of images, the proposed network was trained on an extensive dataset consisting of 50,000 and 10,000 images of 496 × 496 pixels for training and testing respectively.

Feature Pyramid Networks (FPN) [32] is a typical model architecture to generate pyramidal feature representations for object detection. This architecture extracts features at different scales and then fuses them which reportedly provides predictions of higher accuracy [32]. FPN achieved state of the art single-model results on the COCO detection benchmark and has been implemented as a generic feature extractor in several applications such as object detection and instance object segmentation [33,34]. FPN was combined with an hierarchical boosting module to perform pavement crack segmentation obtaining high accuracy and generalizability [35]. The cutting-edge single-stage object detector YOLOv3 adopting FPN was utilized to detect multiple concrete damages of highway bridges [36]. Multiscale feature maps were obtained by a generic pretrained CNN model and fused by implementing FPN in order to apply crack segmentation on concrete images [37].

Furthermore, transfer learning in DL has been extensively implemented on different fields of computer vision with remarkable results and is considered suitable when the training dataset is small allowing for better performance and less computational effort [9,38]. The intuition behind transfer learning for image classification is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world [39]. CNNs utilizing transfer learning have been used extensively for image classification and semantic segmentation in the field of crack detection [19,25,40,41]. Transfer learning was implemented in image-based structural recognition to perform component type identification, spalling condition check, damage level evaluation, and damage type determination [42].

Lately, different studies obtained noteworthy results in crack segmentation by implementing region proposal networks followed by algorithms for pixel-level crack detection [43,44]. In particular, such a hybrid method was proposed by Kang et al. [44] where crack regions detected by Faster R-CNN were processed by a modified tubularity flow field algorithm to segment the crack pixels. As reported by Kang et al. [44], the advantages of this method is that Faster R-CNN can detect crack regions very well even on complex backgrounds while only a dataset of images with bounding boxes of cracks is required which drastically reduces the time to prepare a dataset. As stated by Kang et al. [44], the proposed method is useful for concrete surfaces only and its applicability on different materials might be limited Moreover, Chen et al. [45] implemented an encoder-decoder architecture and proposed a switch method to distinguish between the negative and positive sample automatically and skip the decoder module when the sample is negative to save the inference time.

2.2. Vision-based assessment on masonry surfaces

As shown in Section 2.1, vision-based and, in particular, DL methods for crack detection have been widely applied for concrete surfaces or asphalt. On the contrary, little research has been done on vision-based assessment and specifically for defect detection applied to masonry surfaces. Inarguably, the surface of masonry is less homogeneous and significantly noisier as compared to concrete or asphalt [46]. On top of that, studies have shown that DL models are sensitive to material. In other words, DL models that were trained on a specific surface type failed to achieve same accuracy when the material was different. The development of DL models that could be robustly applied to infrastructure inspection images for both concrete and asphalt pavement was attempted but crack detection models trained on one material did not necessarily work on other materials and significant performance degradation would be expected when testing on other materials [47]. In another study, various CNNs were trained on images from concrete structures for crack detection and their transferability of learned features to photos from different materials was examined by Özgenel and Sorguç [48]. Brickwork images were found to be the most challenging among the tested cases since brickwork jointing and background textures constitute challenging noises. Moreover, it was concluded that the level of variance in the dataset was more important than the number of samples.

Point clouds were obtained with laser scanning and photogrammetry techniques and were combined to detect different types of defects on ashlar masonry walls by using machine learning classification based on geometry and colour information [49]. U-net [27] was used for brick segmentation in masonry walls [50]. McCormick et al. [51] used a system that combined different types of sensors (multiple high-resolution cameras, laser scanning and inertial measurement unit) for tunnel inspection. Digital image correlation techniques were utilized to automatically trace any changes in between consecutive inspections and subsequently an operator would appropriately classify them among a list of defects [51]. Thus, the defect detection process is not fully automatized and human intervention is still required.

CNN to classify different defect types, such as cracking, spalling, mortar loss, and vegetation, from images of masonry structures was used by Brackenbury et al. [46]. In detail, photos were taken from masonry bridges and corrected for perspective distortion and then resized to ensure a constant resolution. Defect classification was implemented on patches of 100 × 100 pixels. It was suggested that by firstly detecting and segmenting mortar joints, and then classifying defects, defected and defect-free areas of the masonry could be all predicted with more confidence and better accuracy.

A novel damage identification architecture to detect two types of damages (efflorescence and spalling) in historic masonry buildings based on the Faster R-CNN ResNet101 model was proposed by [52]. In particular, two orthophotos were extracted from a historical structure and were segmented into small patches of 500 × 500 pixels. The produced patches were annotated with bounding boxes marking the investigated damage types. Quick identification and detection of the surface damage was achieved. The necessity for the expansion of the database with wider range of distances and angles and more types of structural samples was reported.

An automatic vision-based crack detection system using CNN was proposed by Ali et al. [53] to ease the inspection of masonry structures. The feature extraction process was done by CNN from colour images and three classifiers were studied, namely the CNN itself, SVM and Random Forest. False negative areas were found since the system would confuse the grout lines with cracks. Finally, since the cracks on masonry structures could not be easily identified, extreme care was needed when annotating the dataset.

A common limitation observed in the existing literature for vision-based assessment on masonry surfaces is that the developed methods regarded only a single structure and therefore their ability to generalize when tested on more diverse data remains to be evaluated.

3. Dataset preparation

In order to address the lack of data in the literature, a dataset with photos from masonry structures is produced containing complex backgrounds and various crack types and sizes. DL networks are data-driven techniques, thus they heavily rely on the quality and amount of data [54]. Before preparing the masonry dataset for this study, an extensive literature review is performed to spot good and bad practices when collecting data for crack detection. It is highlighted that the goal of training a network is to enhance its ability to generalize when fed with diverse data.

Special care is frequently paid when collecting data so that photos are taken in a homogeneous way keeping constant conditions, such as distance, angle etc. [25,55,56]. Moreover, it is common for datasets for crack detection to be custom-made and manually pre-processed to exclude noisy background and for images to be carefully selected to focus on the cracks [56]. Nevertheless, a common criticism over developed DL methods is that they attain remarkable results when tested on monotonous backgrounds, but their accuracy severely drops when deployed on images with complex backgrounds. Choi and Cha [19] observed that when a CNN trained on images of monotonous background and subsequently tested on a more complex dataset the performance drastically decreased; precision dropped from 0.874 to 0.231. Several studies have emphasized the necessity for more complex datasets [13,57,58]. The issue they raised is particularly important for the context of this paper since masonry surfaces consist of brick or stone materials, possibly with mortar joints, with several complex objects around, such as windows, doors, ornaments, labels, lamps, cables, vegetation etc. which can be characterized as noise for the crack detection process. Other materials, such as concrete or asphalt that crack detection methods have been widely investigated, provide a relatively smooth and flat surface. On the contrary, masonry surface is usually rough and uneven since mortar might protrude around the bricks or some gaps might exist in the interface between mortar joints and brick units. These anomalies might create shadows in the photos especially when the photos are taken with acute angles, causing the network to falsely consider these regions as cracks. Moreover, cracks are usually covered with dust or colour-paints. Therefore, it is deemed that a database as generic as possible would lead to higher chances of developing a tool that is able to perform accurately in real cases.

Taking all these into consideration a masonry dataset is prepared for this study. Photos were collected from different sources. Various images of masonry walls containing cracks were obtained from the Internet. Additionally, photos were taken from different masonry buildings in the Groningen region, The Netherlands. In fact, in order to simulate the scenario where different users will contribute in the data collection by taking photos with devices of different characteristics, various members of our research group were asked to photograph cracks from masonry walls with their phones or DSLR cameras after providing them with simple guidelines. It is noted that photos from masonry surfaces with (Fig. 2a-b) and without (Fig. 2c and d) cracks were taken under similar conditions (angle, distance, etc.) in order to enrich further the non-crack class.

The herein created dataset will be referred to as “masonry dataset”. In total 351 photos containing cracks and 118 without any crack were gathered from masonry surfaces. These photos were divided in patches of 224 × 224 pixels, which leads to 4057 patch containing cracks while extra 7434 non-crack patches were randomly selected from the gathered photos. A sample of photos from the masonry dataset with cracks is presented in Fig. 3. A wide range of scales and resolutions was considered. The crack patches depicture from small (couple of bricks) to larger (whole masonry walls) field of views. Cracks might extend over the joints, the bricks or both. Cracks appearing as straight lines, zigzag or complex shapes were examined. A diverse type of cracks in terms of length, width and shape were included in the masonry dataset. Moreover, the crack patches included different types of noisy background, such as windows, plants, lamps and signs (Fig. 3). Further examples of objects that typically exist around masonry façades and are included in the non-crack patches are shown in Fig. 4.

Along the development of this study, while collecting new data the different networks were run with the available dataset each time. It was observed that the metrics were improving as the masonry dataset was being enriched. The greatest improvement was recorded in the precision value; while extra types of background objects were included in the dataset the easier it was for the networks to learn to accurately negate them. Therefore, by improving how closely the dataset represents the real world the better would be the performance of the networks.

4. Image patch classification for crack detection

4.1. Convolutional neural networks for crack image classification

Image patch classification for crack detection was implemented by leveraging the effect of transfer learning via fine-tuning. The technique of fine-tuning was implemented by discarding the fully connected (FC) layers at the top of a pretrained network and training new, freshly initialized FC layers on the new data with a low learning rate [9]. In detail, a FC layer with 128 features and rectified linear unit (ReLU) activation was added followed by batch normalization and a dropout layer with a probability of 0.5. Batch normalization is a technique that improves the speed, performance, and stability of artificial neural networks and was used to normalize the input layer by adjusting and scaling the activations while dropout temporarily disconnects the neural connections between connected layers during training. Finally, a FC layer with softmax activation was placed to classify the images as crack or non-crack.

Different state of the art CNNs pretrained on ImageNet (1.2 million images with 1000 categories) were examined herein for their efficacy to classify images from masonry surfaces on patch level as crack or non-crack. The considered networks were: VGG16 [59], MobileNet [60], MobileNetV2 [61], InceptionV3 [62], DenseNet121 [63], DenseNet169 [63], ResNet34 [64], ResNet50 [64]. The configuration of ResNet34 and the pre-trained weights were obtained from Yakubovskiy [65], while for the rest of the networks the configuration and the weights were extracted from Keras [66]. The details of the different networks are shown in Table 1. All the models were deposited in the GitHub repository: github.com/dimitrisdais/crack_detection_CNN_masonry.

At this point the architecture of MobileNet is highlighted since it obtained the best results as will be shown below (Section 4.3). MobileNet is a lightweight network destined to run on computationally limited platforms; it achieved accuracy comparable to VGG16 on ImageNet with

Fig. 2. Images from masonry surfaces (a–b) with and (c–d) without cracks.

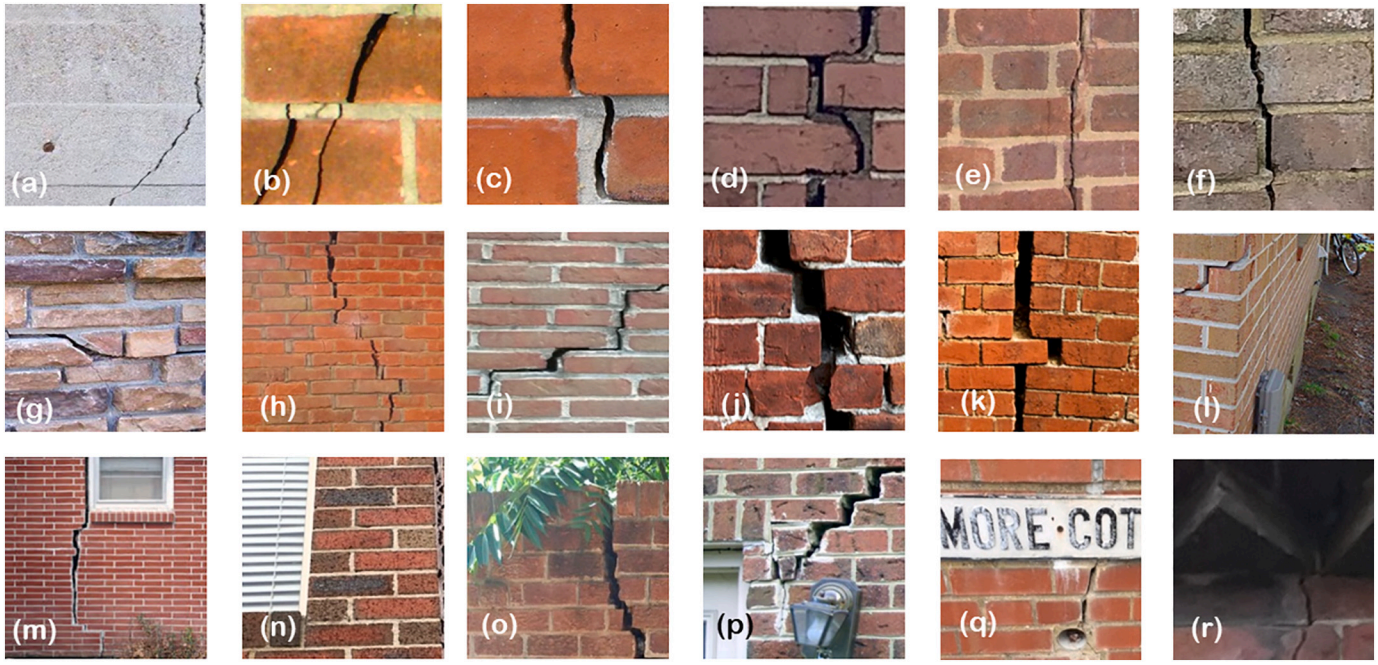

Fig. 3. Images from the masonry dataset containing cracks.

Fig. 4. Images depicturing various 'non-crack' objects included in the masonry dataset for the training of the classification network.

Table 1
Details and metrics of the networks used for image classification. The metrics are presented for the validation set.

Network	Pretrained[a]	Parameters[b] [millions]	Weight[c] [MB]	Accuracy [%]	TN [%]	TP [%]	Analysis Time[d] [hours]	Best Epoch[e]
VGG16	Yes	17.9	70.1	88.0	89.3	85.8	2.2	28
ResNet34	Yes	24.5	96.1	91.6	96.8	82.3	0.7	84
ResNet50	Yes	36.4	142.7	86.5	94.4	72.3	1.1	49
DenseNet121	Yes	13.5	53.5	92.4	95.8	86.2	2.3	48
DenseNet169	Yes	23.1	91.4	89.9	93.3	83.6	2.5	50
InceptionV3	Yes	28.4	111.5	88.4	94.7	77.0	1.7	50
MobileNet	Yes	9.7	37.9	**95.3**	98.4	89.8	1.1	32
	No			89.0	96.4	75.8		95
MobileNetV2	Yes	10.3	40.6	89.7	93.7	82.7	1.2	58

TN: true negatives, TP: true positives.
[a] Whether the encoder of a network is pretrained on ImageNet.
[b] The total number of parameters of a network.
[c] Size of the file where the weights of a network are stored.
[d] Analysis time required to run a network for 50 epochs.
[e] Epoch where the highest accuracy was obtained for the validation set.

only 1/30 of the computational cost and model size [67]. A standard convolution both filters and combines inputs into a new set of outputs in one step. MobileNet is based on depthwise separable convolutions which is a form of factorized convolutions (see Fig. 5); the depthwise convolution applies a single filter to each input channel and the pointwise convolution then applies a 1×1 convolution to combine the outputs of the depthwise convolution. This factorization (Fig. 5) has the effect of drastically reducing computation and model size. MobileNet comprises of multiple factorized layers with depthwise convolution, 1×1 pointwise convolution, batch normalization and ReLU activation (Fig. 6a) instead of layers of regular convolutions followed by batch normalization and ReLU activation (Fig. 6b). The MobileNet architecture has two hyper-parameters that is width and resolution multipliers in order to easily produce smaller versions of the network. Herein, for both hyper-parameters the default value is selected, that is 1, which means than no shrinking is applied to the model [60,66].

MobileNet or networks that made use of depthwise separable convolution have been implemented in recent studies for crack detection. Single Shot MultiBox Detector [68], an object detection framework, was combined with MobileNet to detect different damage types on road surfaces [69]. MobilneNet performed as the encoder of a semantic segmentation network based on DeepLab [18] for real-time tunnel crack analysis [70]. The depthwise separable convolution was used to reduce computational complexity and improve computational efficiency of image classification for crack detection [71]. Depthwise convolutions have been successfully used for pixel-level segmentation of cracks on concrete surfaces [19].

4.2. Training configuration

The networks for image classification are allowed to train for a great number of epochs, with a minimum of 50 epochs, until the accuracy (see Eq. (7)) on the validation set does not increase any further. The data are fed to the network with a batch size of 10.

Optimization in DL networks updates the weight parameters to minimize the loss function. The Adam method (Adaptive Moment Estimation) was found to outperform other stochastic optimization methods [72], i.e. it converges faster, and is selected as the optimizer of the network herein. Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments [72]. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters [72]. The hyperparameters have intuitive interpretations and typically require little tuning. The weight update with Adam optimizer is described as follows:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\, g_t \tag{1}$$

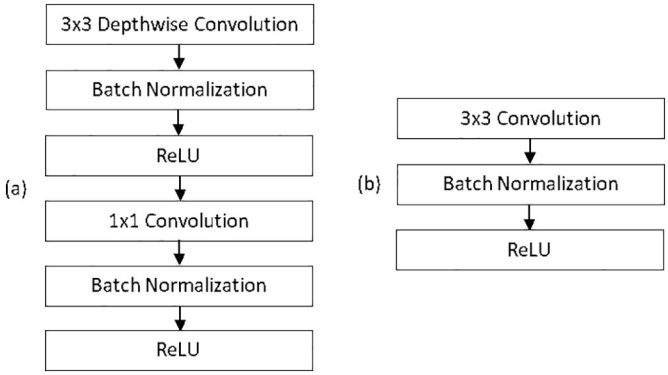

Fig. 6. (a) Factorized layer with depthwise convolution, 1×1 pointwise convolution, batch normalization and ReLU activation. (b) Standard convolutional layer followed by batch normalization and ReLU activation.

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{2}$$

$$\hat{m_t} = \frac{m_t}{1-\beta_1^t} \tag{3}$$

$$\hat{v_t} = \frac{v_t}{1-\beta_2^t} \tag{4}$$

$$w_t = w_{t-1} - \alpha \frac{\hat{m_t}}{\sqrt{\hat{v_t}} + \varepsilon} \tag{5}$$

where t is the timestep, g_t is the gradient vector, m_t and v_t are the first (mean) and second (uncentered variance) biased moment estimates of the gradients respectively, $\hat{m_t}$ and $\hat{v_t}$ are the first (mean) and second (uncentered variance) bias-corrected moment estimates of the gradients respectively, β_1 and β_2 are the exponential decay rates for the moment estimates, α is the learning rate, w is the model weights and $\varepsilon = 10^{-8}$. The default values 0.9 and 0.999 are taken for β_1 and β_2 respectively [66,72]. The networks are trained with a constant learning rate α equal to 0.001.

In the context of an optimization algorithm, a loss function is used to evaluate a candidate solution (i.e. a set of weights) that will minimize the prediction error. The cross entropy (CE) loss function (L_{CE}) is utilized herein and is given as:

$$L_{CE} = -(y log(\hat{y}) + (1-y) log(1-\hat{y}) \tag{6}$$

where y is the ground truth, $\hat{y}$ is the prediction. y can take values equal to 0 (non-crack) or 1 (crack) while $\hat{y}$ can be in the range of 0 to 1.

The performance of the networks is evaluated based on the values of accuracy which is defined as:

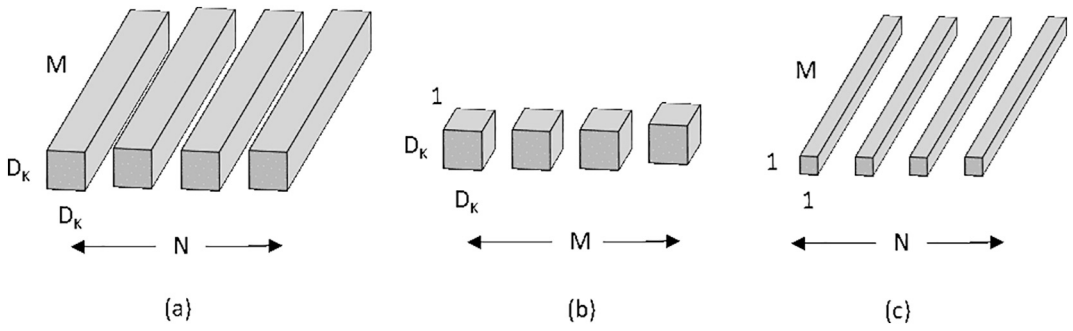

Fig. 5. Schematic representation of (a) standard convolutional filters, (b) depthwise convolutional filters, and (c) 1×1 convolutional filters called pointwise convolution. M, N and D_K stand for the number of input channels, the number of output channels and kernel size respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (7)$$

where *TP*, *TN*, *FP* and *FN* correspond to true positive, true negative, false positive and false negative, respectively. The classification is binary with non-crack and crack cases corresponding to negative and positive class respectively. Thus, TP implies that a crack image is correctly classified while TN means that a non-crack image is predicted accurately. While accuracy performs as an average of the performance of the two classes, TP and TN provide a better insight on the classification error for each class individually and thus are reported as well.

For the training of the image classification networks the 4057 crack and the 7434 non-crack patches of the masonry dataset are used. 60% and 40% of the patches are used for training and validation respectively. The networks are implemented on Keras [66], a high-level neural network API, written in Python and by utilizing TensorFlow as back-end. The networks are run on a laptop with Intel i7 processor with 2.20 GHz, 16 GB RAM and Nvidia GPU GeForce(R) RTX 2060 with 6 GB.

4.3. Results for crack image classification

In this section the results from the trained networks for image classification are presented .The obtained metrics from the trained models on the validation set are enlisted in Table 1 for the epoch that the highest accuracy is reached for each case. While all the considered networks obtain high accuracy on the validation set, that is 88% or more, MobileNet outperforms the rest by scoring accuracy 95.3% (Table 1). In order to examine the benefit of transfer learning, MobileNet is also evaluated without pretraining with its weights randomly initialized [66]. Indeed, the accuracy of MobileNet drops from 95.3% to 89.0% which reveals that transfer learning offers a significant boost to the performance of the network. In more detail, when random initialization is considered, the ratio of TN remains high, that is 96.4%, however TP declines considerably from 89.8% to 75.8%. Consequently, without pretraining the network struggles to differentiate edges corresponding to the crack class and tends to label them as non-crack.

In Figs. 3 and 4 representative images of the masonry dataset are presented. Based on the accuracy of the model it can concluded that the network learns rich features that allow for correct classifications on the dataset produced. A closer look to the performance of MobileNet is highlighted in the produced confusion matrix (Fig. 7). It is inferred that MobileNet excels in predicting correctly the non-crack case with only 1.6% error while the error in the crack class is higher, that is 10.2% of the crack images are classified as non-crack. Different cases of FP and FN predicted with MobileNet from the validation set are displayed in Figs. 8 and 9 respectively. Part of a pipe (Fig. 8a), joints without mortar (Fig. 8b, g–h), edges around doors (Fig. 8e–f), and blurry or dark edges (Fig. 8c–d) are wrongly classified as cracks. Evidently, a further expansion of the masonry dataset should take into consideration a better representation of the cases that yielded FP so that the network will learn their features and correctly classify them. On the other hand, crack images taken with acute angle (Fig. 9a) or with great field of view capturing thin cracks (Fig. 9b–d) are misclassified. Moreover, there are cases of close-up images of thin (Fig. 9e–j) or well-shaped cracks (Fig. 9k–m), crack with missing mortar (Fig. 9n) and crack in dark background (Fig. 9o) that the network falsely negates them to the non-crack class.

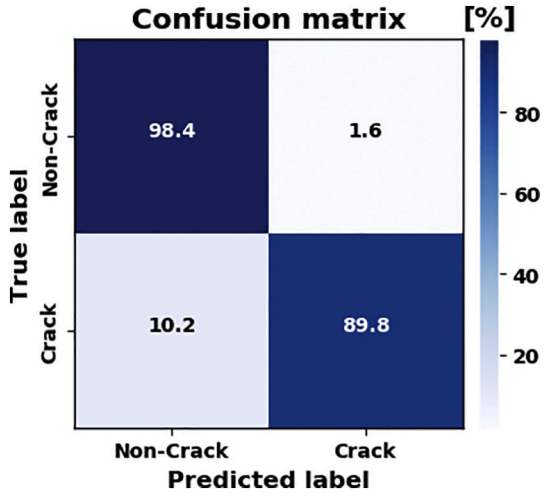

Fig. 7. Confusion matrix obtained with the MobileNet on the validation set.

5. Crack segmentation on pixel level

5.1. Convolutional neural networks for crack segmentation

As per Long et al. [22] "semantic segmentation faces an inherent tension between semantics and location: global information resolves what while local information resolves where". Recently FCNs [22], which are end-to-end networks, have been extensively used for semantic segmentation and in particular for crack segmentation, as highlighted above (Section 2.1). FCNs perform as an extended CNN where the final prediction is an image with semantic segmentation instead of a class identification. FCNs adopt architectures with pyramidal shapes; they follow the usual contracting path (encoder) of image classification networks and replace any FC layers with convolutional layers while on top of the encoder an expanding path (decoder) is added with successive convolutional layers followed by upsampling operators. The encoder captures context while the decoder enables precise localization. In order to avoid loss of low-level information, skip connections are used to allow the decoder to access the low-level features obtained by the encoder branch. A schematic representation of the encoder-decoder architecture of FCNs is shown in Fig. 10. U-net [27], a deep FCN, and FPN [32], a generic pyramid representation, are considered herein and combined with different CNNs performing as the backbone of the encoder part of the network. FPN in fact adopts a similar architecture with U-net, but FPN performs predictions independently at different stages of the expanding path and subsequently concatenates these predictions while U-net only produces predictions at the last stage. The implementation of the U-net and FPN based models with different CNNs as backbone is in accordance with the work of Yakubovskiy [65] and is further elucidated in the next paragraphs. Furthermore, different networks that were successfully used in the literature for crack segmentation are examined in an extensive comparative study.

U-net [27] built upon the original implementation of FCN [22] by increasing the number of feature channels in the upsampling part, which allow the network to propagate context information to higher resolution layers. As a result, in U-net the expansive path is almost symmetric to the contracting path yielding a U-shaped architecture. In the encoder there are repeated blocks of two 3×3 convolutional layers and each of them is followed by batch normalization and ReLU activation. These blocks are referred to as ConvBlock. ConvBlocks are followed by a 2×2 max pooling layer with stride 2 which halves the dimensions of the images and doubles the number of feature channels, a process that is called downsampling. In the decoder, a 2×2 deconvolution layer succeeds each ConvBlock. The deconvolution layer, usually referred to as transpose convolution layer, upsamples the images, meaning it doubles its size and halves the number of feature channels. The final deconvolution layer restores the original size of the image. Then, a 1×1 convolution with sigmoid activation follows which yields the final prediction for each pixel of the image. In total the network has 23 convolutional layers. Same-level ConvBlocks between the encoder and the decoder are merged with skip connections (Fig. 11).

FPN [32] is a typical model architecture to generate pyramidal feature representations for object detection. FPN is independent of the

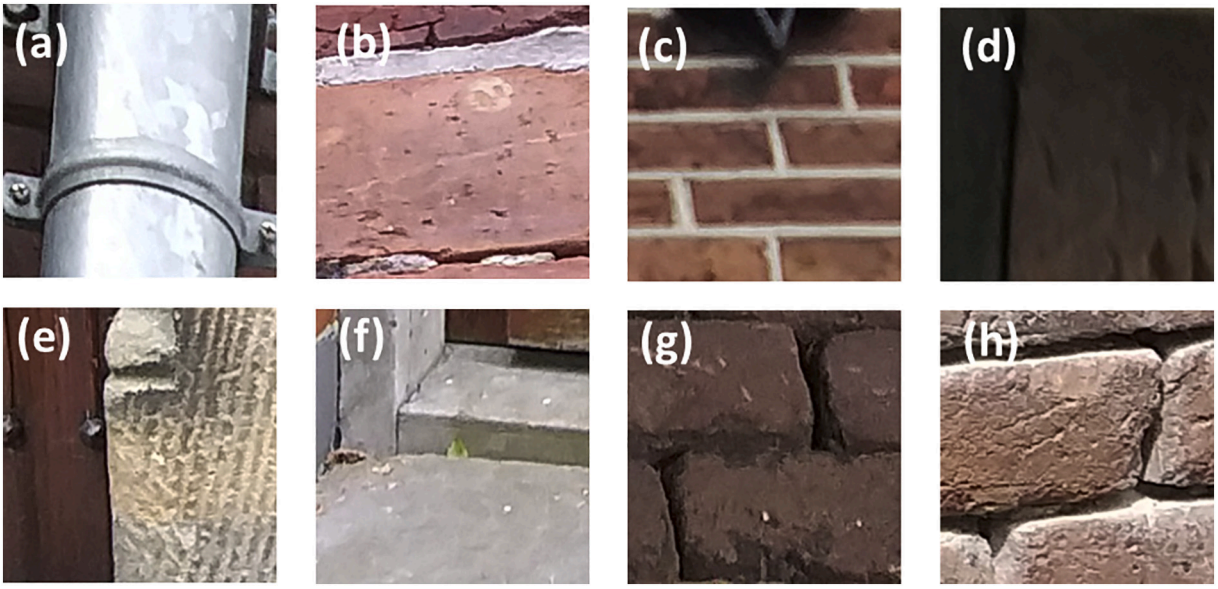

Fig. 8. Non-crack images classified as crack (false positive) by implementing MobileNet on the validation set.

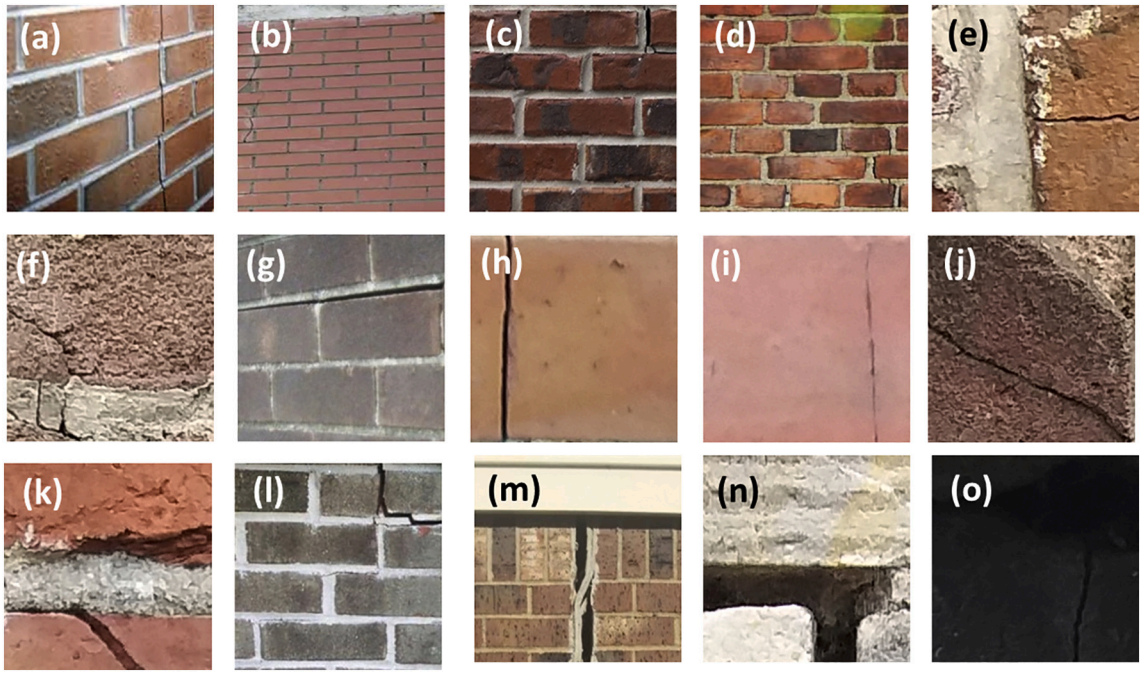

Fig. 9. Crack images classified as non-crack (false negative) by implementing MobileNet on the validation set.

backbone network and its architecture makes it easily configurable to receive different CNNs as the backbone of the encoder. In particular, FPN adopts a convolutional architecture as its backbone, typically designed for image classification, and builds a feature pyramid with a bottom-up pathway, a top-down pathway and lateral connections. The high-level features, which are semantically strong but lower resolution, are upsampled and combined with higher resolution features to generate feature representations that are both high resolution and semantically strong. The upsampling layer repeats the rows and columns of the input features by 2×2 and fills in the new rows and columns by using the nearest neighbour algorithm [66]. The bottom-up pathway which is the feed-forward computation of the backbone CNN produces a feature hierarchy consisting of feature maps at several scales with a scaling step of 2. Layers producing output maps of the same size are considered in the same network stage and for each stage one pyramid level is defined. The top-down pathway obtains higher resolution features by upsampling by a factor of 2 spatially coarser, but semantically stronger, feature maps from higher pyramid levels. These features are then enhanced by element-wise addition with features from the bottom-up pathway which undergo a 1×1 convolutional layer to reduce channel dimensions. Further on, 3×3 convolutions are appended on each merged feature map and the produced maps from the different stages are concatenated. A schematic representation of FPN is displayed in Fig. 12.

The CNNs that were tested for image classification in Section 4 are utilized as the encoder for U-net and FPN in order to perform crack segmentation on pixel level this time. In particular, the considered networks are: VGG16 [59], MobileNet [60], MobileNetV2 [61], InceptionV3 [62], DenseNet121 [63], DenseNet169 [63], ResNet34 [64],

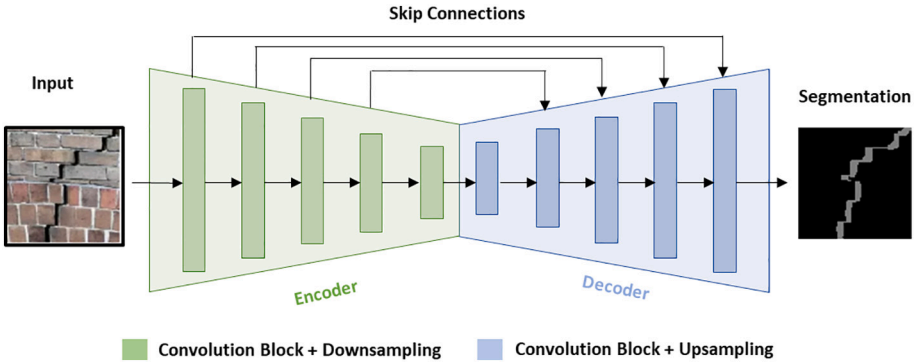

Fig. 10. Schematic representation of the encoder-decoder architecture of Fully Convolutional Networks.

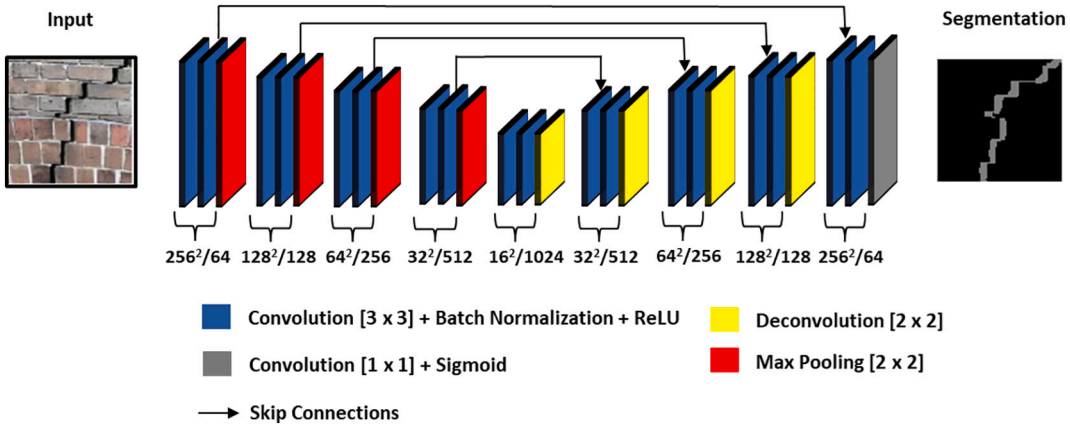

Fig. 11. Illustration of the architecture of U-net as implemented in the herein study. The numbers below the layers denote their size/feature channels respectively.

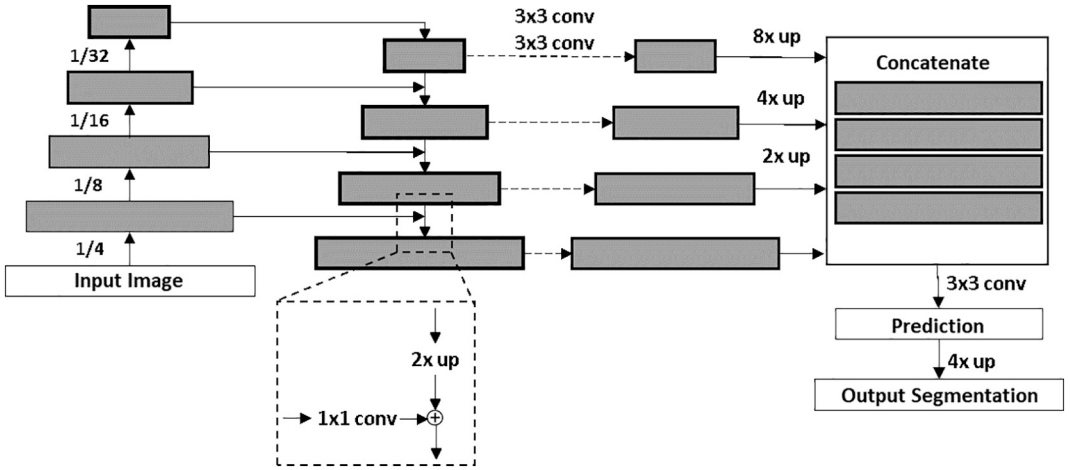

Fig. 12. Schematic representation of Feature Pyramid Network. The rectangles with grey hatch correspond to feature maps and the thicker outlines denote semantically stronger features. The lateral connections and the top-down pathway are merged by addition as shown in the detail (denoted with dashed line). conv and up strand for convolution and upsampling respectively.

ResNet50 [64]. It is noted that U-net is also considered as a standalone network configured as explained above (Fig. 11). For further reference, the models based on U-net and FPN will be called with the base-model followed by the backbone network, e.g. U-net-MobileNet uses U-Net as base-model with MobileNet as backbone. Moreover, apart from U-net, other networks found in the literature and performed well in crack segmentation are examined as well. In particular, DeepLabv3+ [73], DeepCrack [17], and FCN based on VGG16 (will be referred to as FCN-VGG16) [22]. All the networks used in the herein study for segmentation are listed in Table 2 and can be found in the GitHub repository: github.com/dimitrisdais/crack_detection_CNN_masonry.

5.2. Training configuration

The segmentation networks are allowed to train for a great number of epochs, with a minimum of 100 epochs, until the F1 score (see Eq. (11)) on the validation set does not increase any further. The data are fed to the network with a batch size of 4. Similar to the image classification

Table 2
Details and metrics of the networks used for segmentation. The metrics are presented for the validation set.

Network	Pretrained[a]	Loss	Parameters[b] [millions]	Weight[c] [MB]	F1 score [%]	Recall [%]	Precision [%]	Analysis time[d] [hours]	Best Epoch[e]
DeepCrack	No	WCE	29.5	115.5	74.0	80.1	71.6	5.2	28
DeepLabv3+	No	WCE	41.3	162.2	74.9	79.0	73.8	5.6	26
FCN-VGG16	No	WCE	27.8	108.8	75.6	76.6	76.9	2.5	95
U-net	No	WCE	34.5	135.1	75.7	78.9	75.7	5.8	75
U-net-VGG16	Yes	WCE	46.1	180.2	77.2	81.2	76.2	6.0	37
U-net-ResNet34	Yes	WCE	48.0	188.1	77.6	78.3	79.5	4.9	61
U-net-ResNet50	Yes	WCE	73.7	288.5	76.3	80.9	74.8	6.8	45
U-net-Densenet121	Yes	WCE	41.6	163.5	78.1	80.7	78.1	6.2	55
U-net-Densenet169	Yes	WCE	54.3	213.4	78.5	83.5	76.2	7.1	63
U-net-InceptionV3	Yes	WCE	68.5	268.1	77.7	79.2	78.9	6.8	31
U-net-MobileNet	Yes	WCE	37.8	147.9	**79.6**	79.9	81.4	4.8	45
	No	WCE			75.4	80.7	73.4		36
	Yes	CE			76.6	73.0	83.0		36
	Yes	F1-score			78.2	77.1	82.0		29
	Yes	Focal Loss			71.2	61.1	89.4		85
U-net-MobileNetV2	Yes	WCE	39.5	154.9	77.7	76.6	81.9	5.3	58
FPN-VGG16	Yes	WCE	32.2	125.8	77.9	82.0	76.2	5.6	79
FPN-ResNet34	Yes	WCE	38.3	150.2	78.0	81.5	77.2	5.2	36
FPN-ResNet50	Yes	WCE	42.1	164.8	77.2	81.4	75.8	5.8	27
FPN-Densenet121	Yes	WCE	24.6	97.0	79.0	83.6	77.2	6.1	31
FPN-Densenet169	Yes	WCE	30.6	120.8	78.6	80.0	79.5	6.6	59
FPN-InceptionV3	Yes	WCE	40.0	157.2	**79.6**	81.3	80.1	5.7	34
FPN-MobileNet	Yes	WCE	20.8	81.4	79.5	79.5	81.7	4.6	40
FPN-MobileNetV2	Yes	WCE	19.9	78.3	78.5	76.7	82.7	4.8	49

WCE: weighted cross entropy, CE: cross entropy.
[a] Whether the encoder of a network is pretrained on ImageNet.
[b] The total number of parameters of a network.
[c] Size of the file where the weights of a network are stored.
[d] Analysis time required to run a network for 100 epochs.
[e] Epoch where the highest F1 score was obtained for the validation set.

case, the networks for crack segmentation are trained with Adam as optimization algorithm and the learning rate is kept constant equal to 0.0005. The Adam algorithm is further explained above (Section 4.2).

Datasets for image segmentation on crack detection are characterized by severe class imbalance i.e. the background class occupies the greatest part of photos while cracks extend over limited pixels. Due to this imbalance, if special measures are not taken, the network tends to become overconfident in predicting the background class which could lead to misclassifications of cracks and numerous false negatives. To overcome this, the weighted cross entropy (WCE) loss function is implemented herein. In particular, misclassifications of the crack class are penalized with a higher weight. The WCE loss function (L_{wCE}), utilized here, is given as:

$$L_{wCE} = -(\beta\, y log(\hat{y}) + (1-y) log(1-\hat{y})) \tag{8}$$

where y is the ground truth, $\hat{y}$ is the prediction, and β is the weight of the positive class (i.e. crack) chosen as 10. Also, y can take values equal to 0 (background) or 1 (crack); while $\hat{y}$ can be in the range of 0 to 1. In order to evaluate the effect of the loss function to the performance of the network, different loss functions are examined, i.e. CE, F1 score and focal loss. CE and F1 score correspond to the loss functions obtained from Eqs. (6) and (11) respectively while focal loss reshapes CE to down-weight easy examples and thus focus training on hard negatives [74]. It is noted that the focal loss is implemented with the default values suggested by Lin et al. [74].

The evaluation of the network is on the values of precision, recall and F1 score. These metrics are given as:

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

$$F1\ score = \frac{2\ x\ Precision\ x\ Recall}{Precision + Recall} \tag{11}$$

where TP, FP, TN, FN correspond to true positive, false positive, true negative and false negative, respectively.

Another common metric in image segmentation is accuracy which denotes the correctly predicted pixels over the total number of pixels. When there is class imbalance, accuracy is not considered suitable to evaluate the performance of the network since accuracy will yield a score close to its maximum value, that is 1, even if the whole image is classified as the prevailing class (i.e. background). Therefore, accuracy is ignored and is not reported.

Precision regards the correct positive predictions over the total number of the positive predictions and measures the efficiency of the network to negate crack-like objects in the background. Recall considers the completeness of the positive predictions i.e. how many of the actual crack pixels are correctly classified. Precision and recall frequently conflict with each other [75]. In other words, usually high recall values lead to low precisions and vice versa. F1 score is the harmonic mean of precision and recall.

Requesting the model to segment the exact width of the crack has been found to be rather strict and hard to achieve. Different approaches have been implemented in order to overcome this limitation. In particular, connectivity constraints were incorporated in the loss function to take into consideration the relationship among annotations of neighbouring pixels [56]. Other suggested solution was to apply post-processing to isolate noisy parts [76]. A common approach was to allow for some tolerance in the evaluation of the crack detection. Thus, background pixels predicted as cracks (FP) were considered as TP if they were a few pixels apart from the annotated cracks [56,76–78]. The latter proposed approach is followed in the herein study.

For the training of the segmentation networks the 4057 crack patches of the masonry dataset were used. In particular, 60% and 40% of the patches were used for training and validation respectively. The crack

patches were fed to the networks along with pixel-level annotated labels. Similar to the classification networks, the segmentation models were implemented on Keras [66] by utilizing TensorFlow as back-end and were run on the same computing laptop (see Section 4.2 for details).

5.3. Results for crack segmentation

In this section the segmentation results from the trained networks are presented. The obtained metrics from the trained models on the validation set are shown in Table 2 for the epoch that the highest F1 score is reached for each case. From Table 2, a high value of recall does not necessarily mean high precision and vice versa. Thus, F1 score, the average between recall and precision, is deemed the most indicative metric to decide which networks perform better. Thus, U-net-MobileNet and FPN-InceptionV3 attain the highest F1 score, that is 79.6%, and FPN-MobileNet follows with 79.5%.

Firstly, the effect of the loss function on the performance of the networks was evaluated. U-net-MobileNet was trained, apart from WCE, with CE, F1 score and focal loss as loss function. It is noted that similar results were extracted for the other networks but for brevity only results for U-net-MobileNet are presented. The performance of U-net-MobileNet for the different loss functions is displayed in Table 2 and the evolution of the metrics is shown in detail in Fig. 13. As shown in Table 2, the best performance is reached when WCE is utilized; the obtained F1 score is 79.6%, 76.6%, 78.2% and 71.2% for WCE, CE, F1 score and focal loss respectively. Precision is in the range of 90% for CE (Fig. 13b) and focal loss (Fig. 13d) while recall remains significantly lower, i.e. in the range of 60% to 70%. Thus, these two loss functions are not able to handle the class imbalance problem for crack segmentation since the network becomes overconfident in predicting background while neglecting the minority class, that is crack. When WCE (Fig. 13a) and F1 score (Fig. 13c) are used as loss function the discrepancy between precision and recall is less profound. Specifically for WCE, in the first epochs, the recall value ranks approximately 90% while further on converges to 80% and from the 80th epoch onwards decreases to 70%. On the other hand, precision follows an opposite path, starting from 50% and gradually increasing up to 85% in the final epochs. F1 score in the beginning of training is 60% and then converges to value close to slightly below 80%. The highest F1 score is attained in the 45th epoch. The performance of the three metrics indicates that in the beginning, the system is overconfident to predict cracks. In this process, it misclassifies background as cracks. Similar behaviour was reported by [30,56]. As Zhang et al. [75] pointed out, precision and recall frequently conflict with each other and a compromise between recall and precision is made to select the best model. In order to visualize the meaning of different values of recall and precision, predictions with U-net-MobileNet for different images are exhibited for the epochs 3 and 45 which correspond to the highest recall and F1 score respectively (Fig. 14). All the examples in Fig. 14 rank a recall value close to 100% (i.e. maximum value) at epoch 3. Nevertheless, precision and F1 score remain significantly lower. Taking a closer look at the predictions at epoch 3, large parts of the background have been misclassified as cracks (Fig. 14). Regarding the predictions on epoch 45, recall slightly drops while precision significantly increases since the network learns to negate greater parts of the background (Fig. 14).

Furthermore, the networks found in the literature, that is DeepCrack, DeepLabv3+, FCN-VGG16 and U-net, have similar performance in terms of F1 score, i.e. from 74% to 75.7%. U-net outperforms the other networks obtained from the literature achieving F1 score 75.7% with FCN-VGG16 following closely with F1 score 75.6%. Moreover, regarding the performance of the networks found in the literature except for FCN-VGG16, significant discrepancy is observed between the recall and precision values; the networks favour the recall which lead to lower values of precision. The models based on U-net and FPN with a pretrained CNN as backbone attain F1 score from 77.2% to 79.6% which means that they surpass the F1 score, that is 75.7%, of the models found in the literature and are implemented without pretraining. Furthermore, in Table 2 can be observed that U-net and U-net-MobileNet without pretraining reach similar F1 score, that is 75.7% and 75.4% respectively, while the pretrained U-net-MobileNet yields F1 score 79.6%. This observation highlights the effect of pretraining on the performance of the networks; F1 score is boosted by 4.2% when pretraining is considered for U-net-MobileNet. The U-net-MobileNet without pretraining in terms of F1 score records performance similar to FCN-VGG16 and U-net and outperforms DeepCrack and DeepLabv3+. The models based on FPN in general score higher than the corresponding ones built on U-net while the highest F1 score is obtained with U-net-MobileNet and FPN-InceptionV3 (Table 2). It is noted that the models based on FPN have almost half the size of the ones with U-net in terms of model parameters and memory size of the stored weights (Table 2). Thus, FPN models match the performance of the U-net counterparts while being significantly more lightweight networks.

In Fig. 15 different examples from the validation set are presented with predictions obtained with DeepCrack, DeepLabv3+, U-net, U-net-MobileNet (with and without pretraining) and FPN-InceptionV3. In particular, images with edges around openings (Fig. 15a-e), crack-like mortar joints (Fig. 15f-i), shadows (Fig. 15k) and dark spots (Fig. 15l) are displayed. While the pretrained U-net-MobileNet and FPN-InceptionV3 are able to negate different types of noisy background, the rest of the networks (Fig. 15) score lower in terms precision.

Images from the validation set with predictions obtained using U-net-MobileNet have already been presented in Fig. 14 and Fig. 15 while

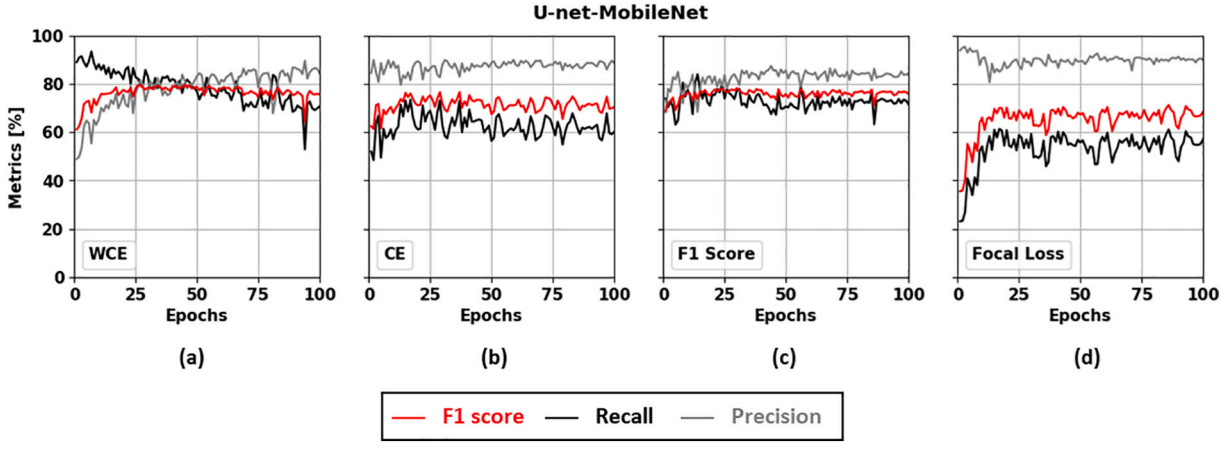

Fig. 13. The metrics F1 score, Recall and Precision as obtained from the pretrained U-net-MobileNet for different loss functions: (a) weighted cross entropy (WCE), (b) cross entropy (CE), (c) F1 score, and (d) focal loss.

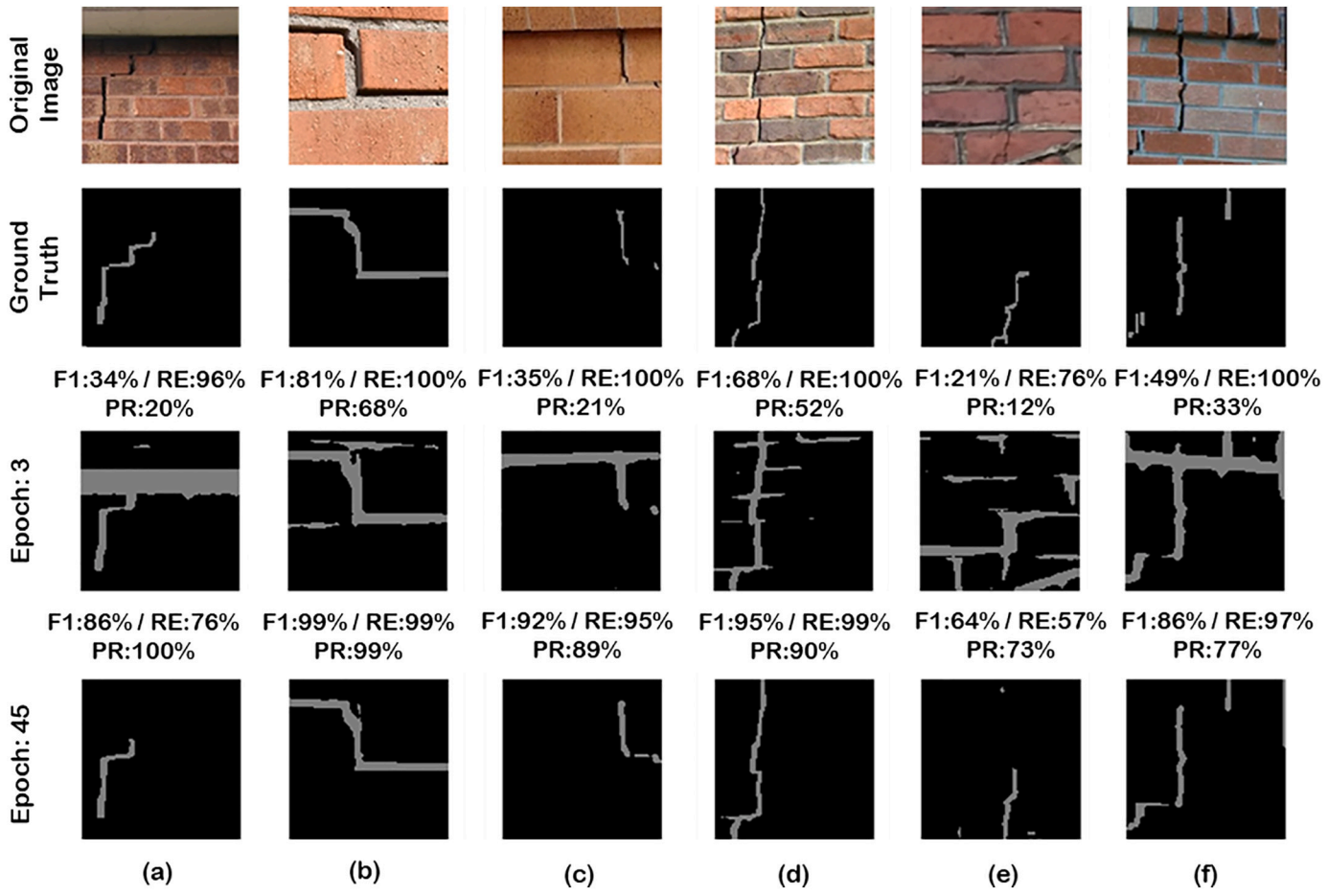

Fig. 14. The original image, the ground truth and the predictions with U-net-MobileNet at epochs 3 and 45 are displayed for different images from the masonry dataset. At the top of each prediction the calculated metrics (F1: F1 score, RE: recall, and PR: precision) are highlighted.

extra examples are shown in Fig. 16. The network successfully segments cracks with different crack size, scale and background complexity; close-up photos (Fig. 16a–c), images with a larger field of view (Fig. 16d–f) and with unwanted objects (i.e. windows and colour-paints) (Fig. 16g–i). Apparently, there are cases that the network failed to perform crack segmentation accurately. For example, in Fig. 16j–k the network fails to detect parts of the cracks. Moreover, Fig. 16l–r displays examples where the model does not manage to negate noisy types of background.

6. Comparative study

In a previous study for crack detection on concrete surfaces it was concluded that when a DL network was trained on images of monotonous background and subsequently tested on a more complex dataset the performance drastically decreased [19]. In more detail, precision from 87.4% fell to 23.1%. Moreover, DL networks trained on concrete images found to perform poorly when tested on masonry images because they are rather complex [48]. This behaviour of CNN was explained by [79]; the transferability of features decreases as the distance between the base task (i.e. training dataset) and target task (i.e. testing dataset) increases. To build up on these findings, U-net-MobileNet trained on the masonry dataset is tested on images from concrete surfaces in order to evaluate the ability of CNNs to generalize over different materials.

In particular, the open source dataset prepared by Yang et al. [24] is selected and will be referred to as the “concrete dataset”. The dataset consists of 776 concrete images containing different crack types. Examples of images in the concrete dataset with their labelled cracks are presented in Fig. 17. An FCN was trained and morphological transformations were applied to further improve the crack segmentation. The reported F1 score, recall and precision were 80%, 79% and 82% respectively [24].

When U-net-MobileNet is tested on the concrete dataset it ranks 74.7%, 70.9%, 91.2% for F1 score, recall and precision respectively. The network does not perform satisfactorily in terms of recall value while excels in terms of precision. These results can be explained by taking a closer look on the predictions on the concrete dataset (Fig. 17). In fact, the network performs exceptionally segmenting cracks with complicated shapes (Fig. 17a–d) obtaining 79% recall or above and a minimum of 94% in terms of precision. On the other hand, the network fails to detect cracks like in Fig. 17e–f but it is noted that these defects look like spalling and do not have a typical crack-like shape; similar defects do not exist in the masonry dataset. Additionally, precision is high which implies that the network can easily negate the background. This could be attributed to the fact that concrete surfaces are rather homogeneous and less complex than masonry surfaces. Consequently, the performance of U-net-MobileNet trained on the masonry dataset deteriorates, i.e. F1 scores declines from 79.6% to 74.7%, when tested on the concrete dataset but not as drastically as reported in the literature when networks trained on concrete images were consequently tested on masonry photos. As explained above (Section 2.2), this is attributed to the fact that masonry surfaces are more complex than concrete ones. It is noted that in the literature there are various datasets of concrete surfaces while only limited data for masonry exist. Thus, when crack segmentation on concrete surfaces is requested, it is recommended to train a model solely on concrete images instead of relying on models trained on masonry

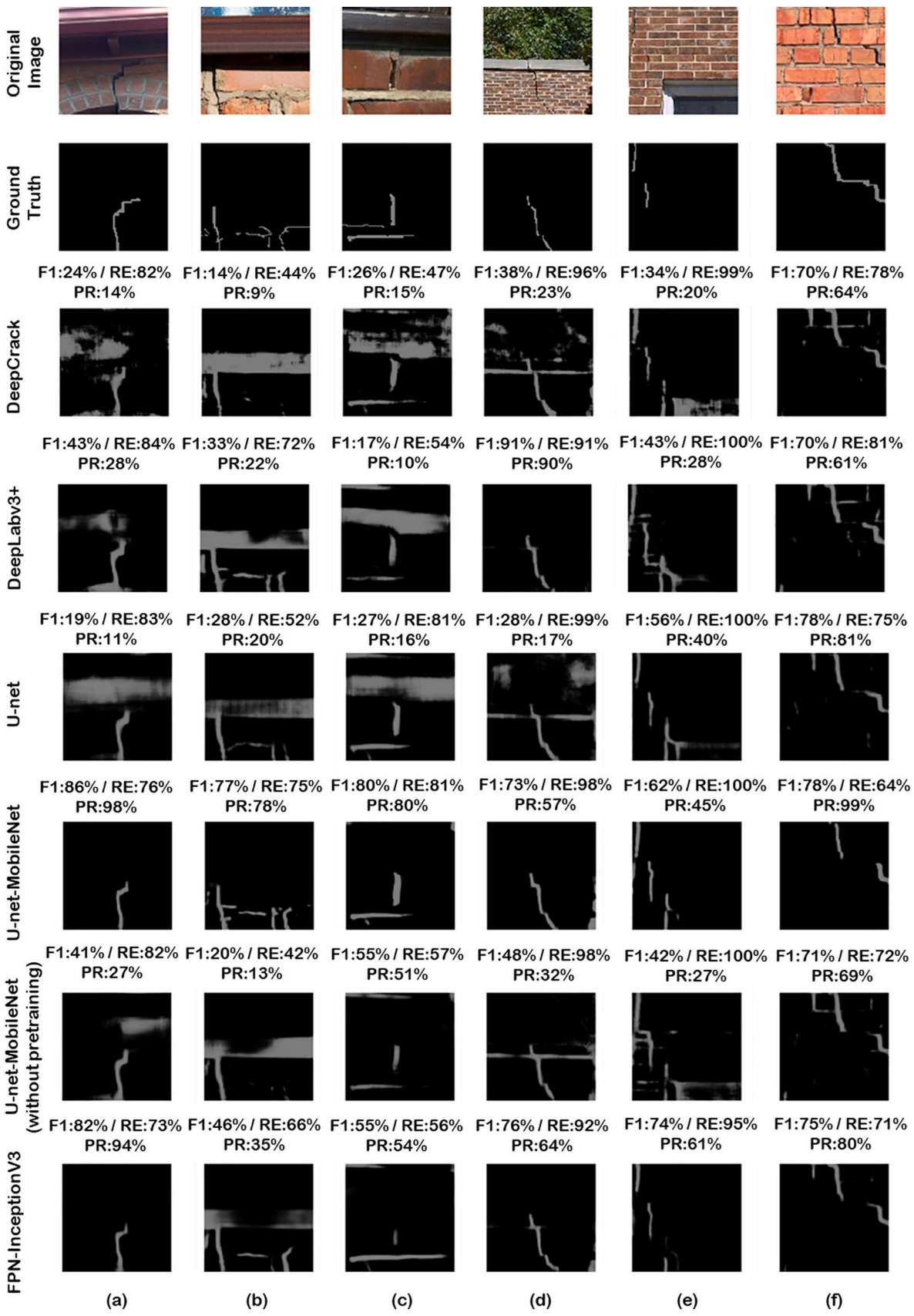

Fig. 15. The original image, the ground truth and the prediction with different netwroks are displayed for different images from the masonry dataset. At the top of each prediction the calculated metrics (F1: F1 score, RE: recall, and PR: precision) are highlighted.

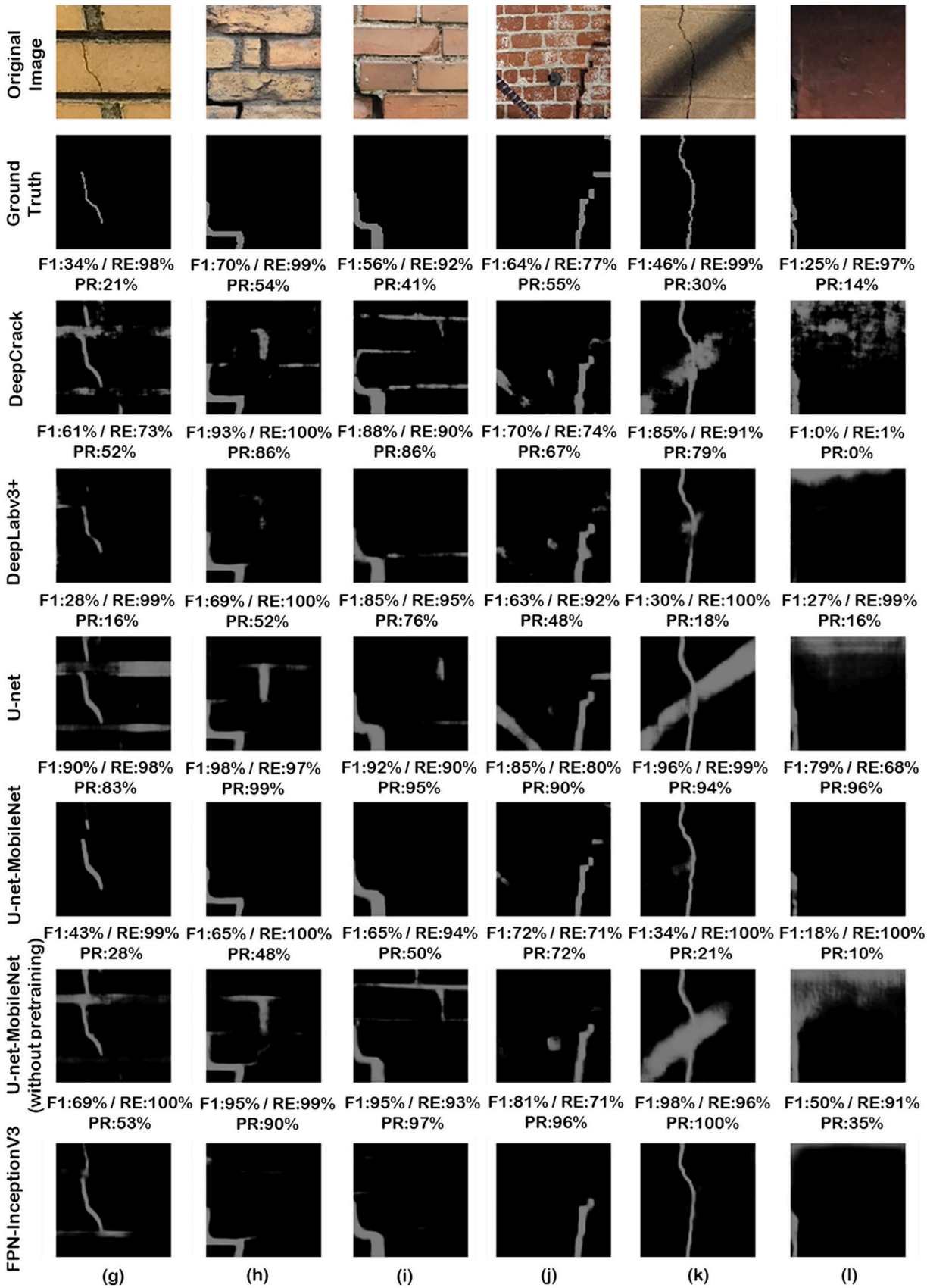

Fig. 15. (*continued*).

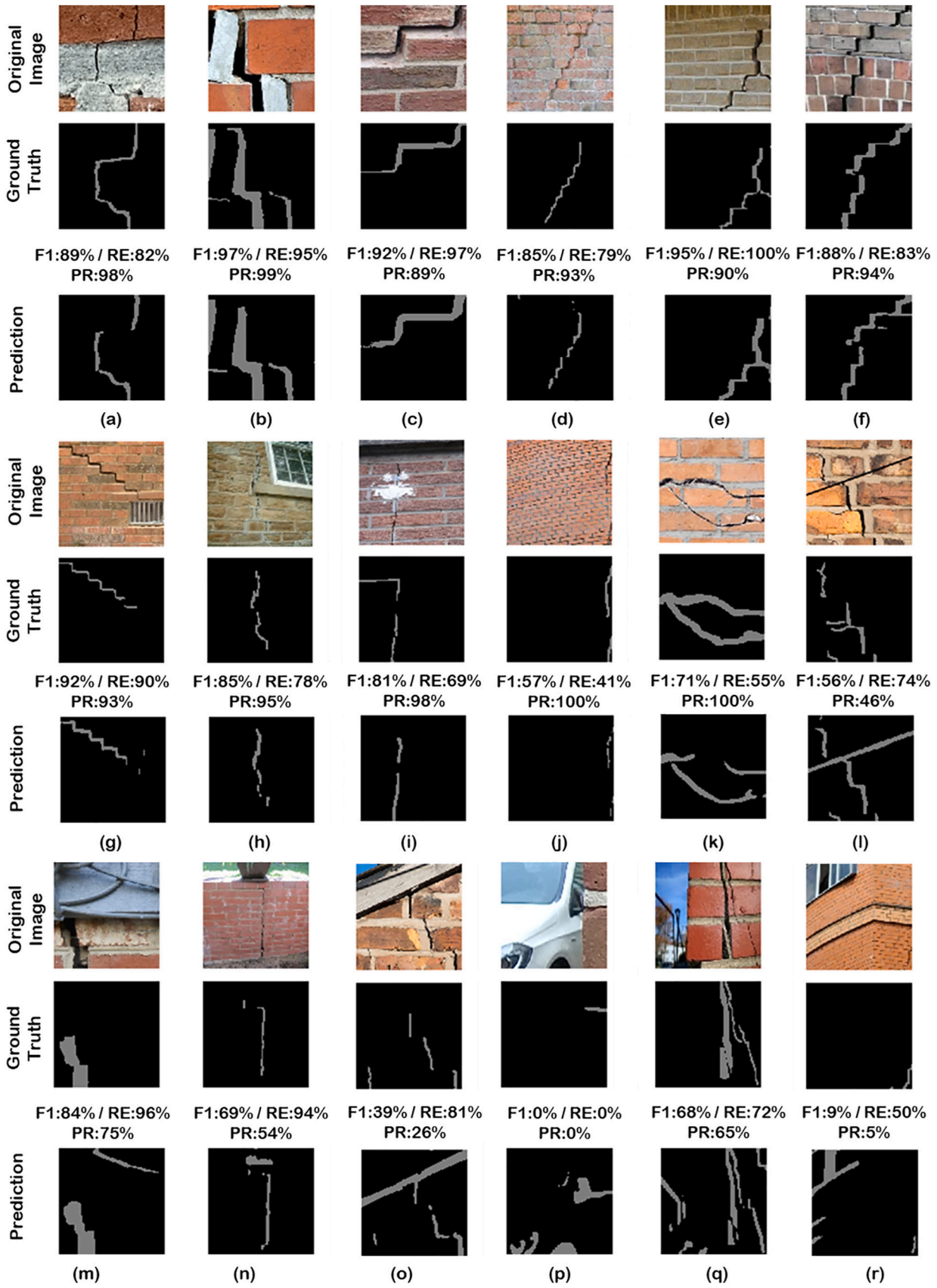

Fig. 16. The original image, the ground truth and the prediction with U-net-MobileNet are displayed for different images from the masonry dataset. At the top of each prediction the calculated metrics (F1: F1 score, RE: recall, and PR: precision) are highlighted.

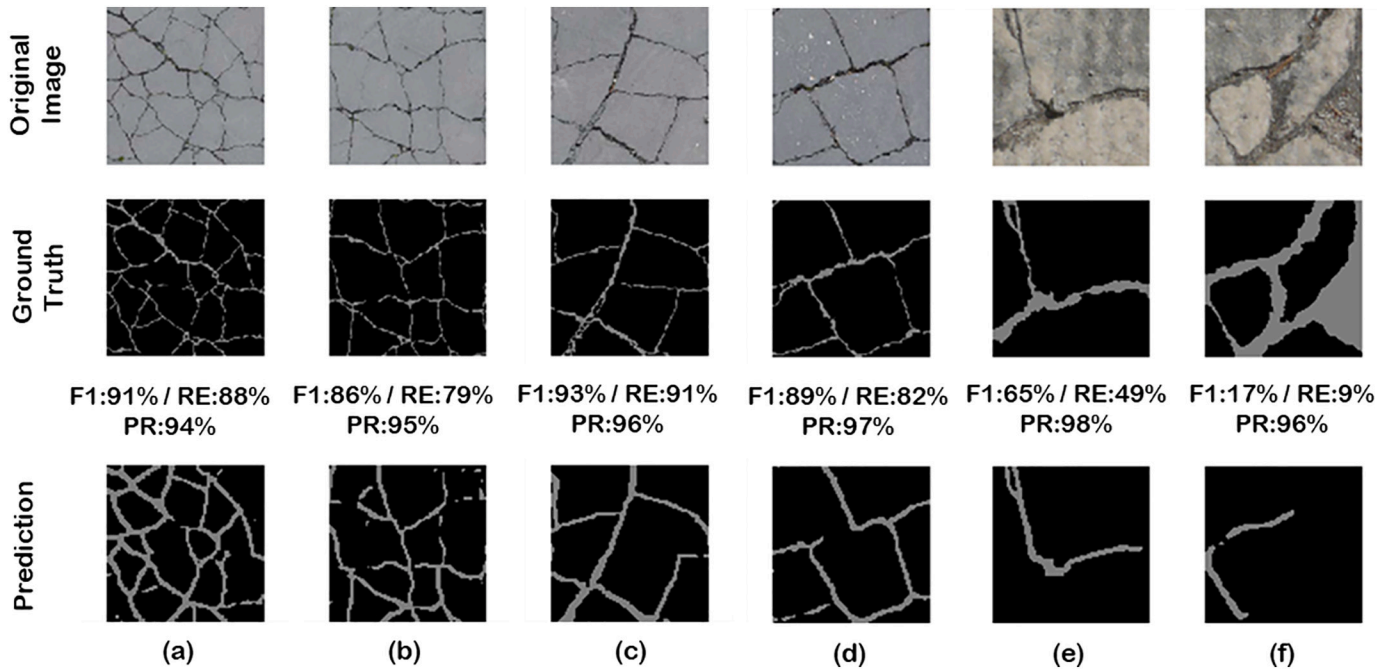

Fig. 17. Results obtained when U-net-MobileNet is trained on the masonry dataset and tested on photos from the concrete dataset [24]. For each image, the original image, the ground truth and the prediction are displayed. At the top of each prediction the calculated metrics (F1: F1 score, RE: recall, and PR: precision) are highlighted.

data. On the other hand, for cases where only few data exist, e.g. timber surfaces, a model trained on a dataset with complex backgrounds like the masonry dataset produced herein could be an alternative.

7. Conclusions

In this study the feasibility of DL techniques for crack detection on images from masonry walls is investigated. Even though masonry surfaces have been reported to be rather complex for CNN on crack detection, this study showcases that DL algorithms are able to accurately detect cracks from images of real masonry surfaces. In order to address the lack of data in the literature, a dataset with photos from masonry structures was produced containing complex backgrounds and various crack types and sizes. Different DL networks are considered and by leveraging the effect of transfer learning crack detection on masonry surfaces is performed both on patch and pixel level. To the authors' best knowledge, this is the first implementation of DL for pixel-level crack segmentation on masonry surfaces. State of the art CNNs pretrained on ImageNet are examined for their efficacy to classify images from masonry surfaces on patch level with MobileNet obtaining the highest accuracy, that is 95.3%. U-net, a deep FCN, and FPN, a generic pyramid representation, are combined with different pretrained CNNs performing as the backbone of the encoder part of the network to perform pixel level crack segmentation. U-net-MobileNet and FPN-InceptionV3 attain the highest F1 score, that is 79.6%, and outperform other networks for crack segmentation from the literature. In particular, for U-net-MobileNet, when the backbone CNN is considered without pretraining, F1 score declines from 79.6% to 75.4%, which demonstrates the beneficial effect of transfer learning. The ability of CNNs to generalize over different materials is evaluated. The performance of U-net-MobileNet trained on the masonry dataset deteriorates, i.e. F1 scores declines from 79.6% to 74.7%, when tested on concrete images but not as drastically as reported in the literature when networks trained on concrete images were consequently tested on masonry photos. Codes, data and networks relevant to the herein study can be found in the GitHub repository: github.com/dimitrisdais/crack_detection_CNN_masonry.

Although the proposed DL algorithms achieved promising results, further improvements are required to achieve a fully automated vision-based assessment of masonry surfaces. The current study focuses on the detection of cracks but in the future the annotations of the masonry dataset could be updated to consider other defect types as well. The results of DL methods heavily rely on the quality of data. Thus, the expansion of the current masonry dataset is highly recommended with special care for the inclusion of even broader background types. In particular, including photos under low-lighting conditions and further evaluating the accuracy of the crack detection is highly recommended. With the increasing accessibility to high quality camera sensors it is advised that the research community develops ways to further mobilize engineers, practitioners and citizens to contribute in the data collection process and provide them with guidelines and automatic procedures that will render the gathered data reliable. Significant research has been devoted to the automatic semantic segmentation of photos coming from building façades, a technique known as façade parsing. Further studies are advised to evaluate whether façade parsing could be utilized to preliminarily detect objects like doors, ornaments, etc. and negate them so that the network would search for defects only on masonry surfaces. Herein, networks based on U-net and FPN architectures were implemented. Recent studies have come up with updated versions of these architectures which outperformed the original implementations. A further investigation whether these updated versions could improve the accuracy of the herein suggested DL algorithms for crack detection is encouraged. The best performing networks implemented herein scored better than other networks which have already been successfully used in the literature for crack segmentation on concrete or asphalt surfaces. Thus, it is highly recommended that the best architectures used herein are implemented on other types of surfaces as future research.

Declaration of Competing Interest

None.

Acknowledgements

Several photos obtained by inspectors for Helifix, UK, and were

kindly offered to expand our masonry dataset and their contribution to this study is highly appreciated. Nektarios Lianos, Computer Vision specialist at Geomagical Labs, is acknowledged for his insightful comments on the implementation of the deep learning networks. The work has been partially funded by RVO within the project “SafeGO - Seismic Monitoring, Design And Strengthening For thE GrOningen Region”, Grant No: RAAK.MKB09.021. The crack detection method was developed alongside the project “Seismic Monitoring of Historical Buildings in Groningen” funded by Rijksdienst voor het Cultureel Erfgoed, Grant No: 126761.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

References

[1] M. Tomazevic, Earthquake-Resistant Design of Masonry Buildings, Imperial College Press, 1999, https://doi.org/10.1142/p055. ISBN: 1-86094-066-8.
[2] D. Dais, V. Sarhosis, E. Smyrou, İ.E. Bal, R. Coningham, A. Joshi, K. Acharya, P. Maskey, K. Weise, R. Kunwar, Investigations on the restoration and seismic enhancement options for the Jaisedewal Temple after the Gorkha earthquake in Nepal, in: SECED 2019 Conference, Greenwich, UK, 2019. http://www.seced.org.uk/ (accessed October 1, 2019).
[3] İ.E. Bal, D. Dais, E. Smyrou, V. Sarhosis, Monitoring of a historical masonry structure in case of induced seismicity, Int. J. Architect. Herit. (2020) 1–18, https://doi.org/10.1080/15583058.2020.1719230.
[4] M. Gilbert, Fatigue in railway bridges, in: M. Robinson, A. Kapoor (Eds.), Fatigue in Railway Infrastructure, Elsevier, 2009, pp. 58–95, https://doi.org/10.1533/9781845697020.58.
[5] B.M. Phares, G.A. Washer, D.D. Rolander, B.A. Graybeal, M. Moore, Routine highway bridge inspection condition documentation accuracy and reliability, J. Bridg. Eng. 9 (4) (2004) 403–413, https://doi.org/10.1061/(ASCE)1084-0702(2004)9:4(403).
[6] D.F. Laefer, J. Gannon, E. Deely, Reliability of crack detection methods for baseline condition assessments, J. Infrastruct. Syst. 16 (2) (2010) 129–137, https://doi.org/10.1061/(ASCE)1076-0342(2010)16:2(129).
[7] B.F. Spencer, V. Hoskere, Y. Narazaki, Advances in computer vision-based civil infrastructure inspection and monitoring, Engineering 5 (2) (2019) 199–222, https://doi.org/10.1016/j.eng.2018.11.030.
[8] Z.-Q. Zhao, P. Zheng, S.-T. Xu, X. Wu, Object detection with deep learning: a review, IEEE Trans. Neural Netw. Learn. Syst. 30 (11) (2019) 3212–3232, https://doi.org/10.1109/TNNLS.2018.2876865.
[9] A. Rosebrock, Deep Learning for Computer Vision with Python. www.pyimagesearch.com, 2017 (accessed October 1, 2020).
[10] I.-H. Kim, H. Jeon, S.-C. Baek, W.-H. Hong, H.-J. Jung, Application of crack identification techniques for an aging concrete bridge inspection using an unmanned aerial vehicle, Sensors 18 (6) (2018) 1881, https://doi.org/10.3390/s18061881.
[11] M. Mohtasham Khani, S. Vahidnia, L. Ghasemzadeh, Y.E. Ozturk, M. Yuvalaklioglu, S. Akin, N.K. Ure, Deep-learning-based crack detection with applications for the structural health monitoring of gas turbines, Struct. Health Monit. 19 (5) (2020) 1440–1452, https://doi.org/10.1177/1475921719883202.
[12] A. Zhang, K.C.P. Wang, Y. Fei, Y. Liu, S. Tao, C. Chen, J.Q. Li, B. Li, Deep learning–based fully automated pavement crack detection on 3D asphalt surfaces with an improved CrackNet, J. Comput. Civ. Eng. 32 (5) (2018), 04018041, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000775.
[13] J. Zhang, C. Lu, J. Wang, L. Wang, X.-G. Yue, Concrete cracks detection based on FCN with dilated convolution, Appl. Sci. 9 (13) (2019) 2686, https://doi.org/10.3390/app9132686.
[14] C. Feng, H. Zhang, S. Wang, Y. Li, H. Wang, F. Yan, Structural damage detection using deep convolutional neural network and transfer learning, KSCE J. Civ. Eng. 23 (10) (2019) 4493–4502, https://doi.org/10.1007/s12205-019-0437-z.
[15] Y.-J. Cha, W. Choi, O. Büyüköztürk, Deep learning-based crack damage detection using convolutional neural Networks, Comput. Aided Civil Infrastruct. Eng. 32 (5) (2017) 361–378, https://doi.org/10.1111/mice.12263.
[16] Y.J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, O. Büyüköztürk, Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types, Comput. Aided Civil Infrastruct. Eng. 33 (9) (2018) 731–747, https://doi.org/10.1111/mice.12334.
[17] Y. Liu, J. Yao, X. Lu, R. Xie, L. Li, DeepCrack: a deep hierarchical feature learning architecture for crack segmentation, Neurocomputing 338 (2019) 139–153, https://doi.org/10.1016/j.neucom.2019.01.036.
[18] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs, IEEE Trans. Pattern Anal. Mach. Intell. 40 (4) (2018) 834–848, https://doi.org/10.1109/TPAMI.2017.2699184.
[19] W. Choi, Y.-J. Cha, SDDNet: real-time crack segmentation, IEEE Trans. Ind. Electron. 67 (9) (2020) 8016–8025, https://doi.org/10.1109/TIE.2019.2945265.
[20] Y. Li, H. Li, H. Wang, Pixel-wise crack detection using deep local pattern predictor for robot application, Sensors 18 (9) (2018) 3042, https://doi.org/10.3390/s18093042.
[21] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, J. Garcia-Rodriguez, A review on deep learning techniques applied to semantic segmentation, ArXiv (2017). http://arxiv.org/abs/1704.06857. (Accessed 1 October 2020).
[22] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Boston, MA, USA, 2015, pp. 3431–3440, https://doi.org/10.1109/CVPR.2015.7298965.
[23] C.V. Dung, L.D. Anh, Autonomous concrete crack detection using deep fully convolutional neural network, Autom. Constr. 99 (2019) 52–58, https://doi.org/10.1016/j.autcon.2018.11.028.
[24] X. Yang, H. Li, Y. Yu, X. Luo, T. Huang, X. Yang, Automatic pixel-level crack detection and measurement using fully convolutional network, Comput. Aided Civil Infrastruct. Eng. 33 (12) (2018) 1090–1109, https://doi.org/10.1111/mice.12412.
[25] S. Li, X. Zhao, G. Zhou, Automatic pixel-level multiple damage detection of concrete structure using fully convolutional network, Comput. Aided Civil Infrastruct. Eng. 34 (7) (2019) 616–634, https://doi.org/10.1111/mice.12433.
[26] V. Hoskere, Y. Narazaki, T.A. Hoang, B.F. Spencer, MaDnet: multi-task semantic segmentation of multiple types of structural materials and damage in images of civil infrastructure, J. Civ. Struct. Heal. Monit. 10 (5) (2020) 757–773, https://doi.org/10.1007/s13349-020-00409-0.
[27] O. Ronneberger, P. Fischer, T. Brox, U-net: convolutional networks for biomedical image segmentation, Med. Image Comput. Comput. Assist.Interv. (MICCAI) 9351 (2015) 234–241, https://doi.org/10.1007/978-3-319-24574-4_28.
[28] J. Konig, M. David Jenkins, P. Barrie, M. Mannion, G. Morison, A convolutional neural network for pavement surface crack segmentation using residual connections and attention gating, in: 2019 IEEE International Conference on Image Processing (ICIP), IEEE, Taipei, Taiwan, 2019, pp. 1460–1464, https://doi.org/10.1109/ICIP.2019.8803060.
[29] M. David Jenkins, T.A. Carr, M.I. Iglesias, T. Buggy, G. Morison, A deep convolutional neural network for semantic pixel-wise segmentation of road and pavement surface cracks, in: 2018 26th European Signal Processing Conference (EUSIPCO), IEEE, Rome, Italy, 2018, pp. 2120–2124, https://doi.org/10.23919/EUSIPCO.2018.8553280.
[30] Z. Liu, Y. Cao, Y. Wang, W. Wang, Computer vision-based concrete crack detection using U-net fully convolutional networks, Autom. Constr. 104 (2019) 129–139, https://doi.org/10.1016/j.autcon.2019.04.005.
[31] Li Ma, He Ren, Liu, Automatic tunnel crack detection based on U-net and a convolutional neural network with alternately updated clique, Sensors 20 (3) (2020) 717, https://doi.org/10.3390/s20030717.
[32] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Honolulu, HI, USA, 2017, pp. 936–944, https://doi.org/10.1109/CVPR.2017.106.
[33] K. He, G. Gkioxari, P. Dollar, R. Girshick, Mask R-CNN, in: 2017 IEEE International Conference on Computer Vision (ICCV), IEEE, Venice, Italy, 2017, pp. 2980–2988, https://doi.org/10.1109/ICCV.2017.322.
[34] J. Redmon, A. Farhadi, YOLOv3: an incremental improvement, ArXiv (2018). http://arxiv.org/abs/1804.02767 (accessed October 1, 2020).
[35] F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, H. Ling, Feature pyramid and hierarchical boosting network for pavement crack detection, IEEE Trans. Intell. Transp. Syst. 21 (4) (2020) 1525–1535, https://doi.org/10.1109/TITS.2019.2910595.
[36] C. Zhang, C.C. Chang, M. Jamshidi, Bridge damage detection using a single-stage detector and field inspection images, ArXiv (2018). http://arxiv.org/abs/1812.10590 (accessed October 1, 2020).
[37] F. Ni, J. Zhang, Z. Chen, Pixel-level crack delineation in images with convolutional feature fusion, Struct. Control. Health Monit. 26 (1) (2019), e2286, https://doi.org/10.1002/stc.2286.
[38] F. Chollet, Deep Learning with Python, Manning Publications Co., 2017. ISBN: 9781617294433.
[39] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: large-scale machine learning on heterogeneous distributed systems, ArXiv (2016). http://arxiv.org/abs/1603.04467 (accessed October 1, 2020).
[40] Q. Yang, W. Shi, J. Chen, W. Lin, Deep convolution neural network-based transfer learning method for civil infrastructure crack detection, Autom. Constr. 116 (2020) 103199, https://doi.org/10.1016/j.autcon.2020.103199.
[41] S. Bang, S. Park, H. Kim, H. Kim, Encoder–decoder network for pixel-level road crack detection in black-box images, Comput. Aided Civil Infrastruct. Eng. 34 (8) (2019) 713–727, https://doi.org/10.1111/mice.12440.
[42] Y. Gao, K.M. Mosalam, Deep transfer learning for image-based structural damage recognition, Comput. Aided Civil Infrastruct. Eng. 33 (9) (2018) 748–768, https://doi.org/10.1111/mice.12363.

[43] R. Kalfarisi, Z.Y. Wu, K. Soh, Crack detection and segmentation using deep learning with 3D reality mesh model for quantitative assessment and integrated visualization, J. Comput. Civ. Eng. 34 (3) (2020), 04020010, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000890.
[44] D. Kang, S.S. Benipal, D.L. Gopal, Y. Cha, Hybrid pixel-level concrete crack segmentation and quantification across complex backgrounds using deep learning, Autom. Constr. 118 (2020) 103291, https://doi.org/10.1016/j.autcon.2020.103291.
[45] H. Chen, H. Lin, M. Yao, Improving the efficiency of encoder-decoder architecture for pixel-level crack detection, IEEE Access. 7 (2019) 186657–186670, https://doi.org/10.1109/ACCESS.2019.2961375.
[46] D. Brackenbury, I. Brilakis, M. DeJong, Automated Defect Detection For Masonry Arch Bridges, International Conference on Smart Infrastructure and Construction 2019 (ICSIC), ICE Publishing, 2019, pp. 3–9, https://doi.org/10.1680/icsic.64669.003.
[47] M. Alipour, D.K. Harris, Increasing the robustness of material-specific deep learning models for crack detection across different materials, Eng. Struct. 206 (2020) 110157, https://doi.org/10.1016/j.engstruct.2019.110157.
[48] Ç.F. Özgenel, A.G. Sorguç, Performance comparison of pretrained convolutional neural networks on crack detection in buildings, in: J. Teizer (Ed.), Proceedings of the 35th International Symposium on Automation and Robotics in Construction (ISARC), Berlin, Germany, 2018, pp. 693–700, https://doi.org/10.22260/ISARC2018/0094.
[49] E. Valero, A. Forster, F. Bosché, E. Hyslop, L. Wilson, A. Turmel, Automated defect detection and classification in ashlar masonry walls using machine learning, Autom. Constr. 106 (2019) 102846, https://doi.org/10.1016/j.autcon.2019.102846.
[50] Y. Ibrahim, B. Nagy, C. Benedek, CNN-based watershed marker extraction for brick segmentation in masonry walls, in: F. Karray, A. Campilho, A. Yu (Eds.), Image Analysis and Recognition. ICIAR 2019, Springer, Cham, 2019, pp. 332–344, https://doi.org/10.1007/978-3-030-27202-9_30.
[51] A. Najimi, D. Jonas, N.J. McCormick, S.A. Kimkeran, Assessing the condition of railway assets using DIFCAM: Results from tunnel examinations, in: 6th IET Conference on Railway Condition Monitoring (RCM 2014), Institution of Engineering and Technology, Birmingham, UK, 2014, pp. 1–6, https://doi.org/10.1049/cp.2014.1002.
[52] N. Wang, X. Zhao, P. Zhao, Y. Zhang, Z. Zou, J. Ou, Automatic damage detection of historic masonry buildings based on mobile deep learning, Autom. Constr. 103 (2019) 53–66, https://doi.org/10.1016/j.autcon.2019.03.003.
[53] L. Ali, Damage detection and localization in masonry structure using faster region convolutional networks, Int. J. GEOMATE 17 (59) (2019) 98–105, https://doi.org/10.21660/2019.59.8272.
[54] K. Zhang, Y. Zhang, H.-D. Cheng, CrackGAN: pavement crack detection using partially accurate ground truths based on generative adversarial learning, in: IEEE Transactions on Intelligent Transportation Systems, 2020, pp. 1–14, https://doi.org/10.1109/TITS.2020.2990703.
[55] S. Dorafshan, R.J. Thomas, M. Maguire, Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete, Constr. Build. Mater. 186 (2018) 1031–1045, https://doi.org/10.1016/j.conbuildmat.2018.08.011.
[56] Q. Mei, M. Gül, M.R. Azim, Densely connected deep neural network considering connectivity of pixels for automatic crack detection, Autom. Constr. 110 (2020) 103018, https://doi.org/10.1016/j.autcon.2019.103018.
[57] M. Alipour, D.K. Harris, G.R. Miller, Robust pixel-level crack detection using deep fully convolutional neural Networks, J. Comput. Civ. Eng. 33 (6) (2019), 04019040, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000854.
[58] D. Tabernik, S. Šela, J. Skvarč, D. Skočaj, Segmentation-based deep-learning approach for surface-defect detection, J. Intell. Manuf. 31 (3) (2020) 759–776, https://doi.org/10.1007/s10845-019-01476-x.
[59] S. Liu, W. Deng, Very deep convolutional neural network based image classification using small training sample size, in: 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), IEEE, San Diego, USA, 2015, pp. 730–734, https://doi.org/10.1109/ACPR.2015.7486599.
[60] 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, ArXiv (2017). http://arxiv.org/abs/1704.04861 (accessed October 1, 2020).
[61] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, MobileNetV2: inverted residuals and linear bottlenecks, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, Salt Lake City, UT, USA, 2018, pp. 4510–4520, https://doi.org/10.1109/CVPR.2018.00474.
[62] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Las Vegas, NV, USA, 2016, pp. 2818–2826, https://doi.org/10.1109/CVPR.2016.308.
[63] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Honolulu, HI, USA, 2017, pp. 2261–2269, https://doi.org/10.1109/CVPR.2017.243.
[64] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Las Vegas, NV, USA, 2016, pp. 770–778, https://doi.org/10.1109/CVPR.2016.90.
[65] P. Yakubovskiy, Segmentation Models, GitHub. https://github.com/qubvel/segmentation_models, 2019 (accessed October 1, 2020).
[66] F. Chollet, Keras. https://keras.io, 2015 (accessed October 1, 2020).
[67] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, K. Murphy, Speed/accuracy trade-offs for modern convolutional object detectors, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Honolulu, HI, USA, 2017, pp. 3296–3297, https://doi.org/10.1109/CVPR.2017.351.
[68] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: Single Shot MultiBox Detector, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, Springer, Cham, 2016, pp. 21–37, https://doi.org/10.1007/978-3-319-46448-0_2.
[69] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, H. Omata, Road damage detection and classification using deep neural networks with smartphone images, Comput. Aided Civil Infrastruct. Eng. 33 (12) (2018) 1127–1141, https://doi.org/10.1111/mice.12387.
[70] Q. Song, Y. Wu, X. Xin, L. Yang, M. Yang, H. Chen, C. Liu, M. Hu, X. Chai, J. Li, Real-time tunnel crack analysis system via deep learning, IEEE Access 7 (2019) 64186–64197, https://doi.org/10.1109/ACCESS.2019.2916330.
[71] H. Xu, X. Su, Y. Wang, H. Cai, K. Cui, X. Chen, Automatic bridge crack detection using a convolutional neural network, Appl. Sci. 9 (14) (2019) 2867, https://doi.org/10.3390/app9142867.
[72] D.P. Kingma, J.L. Ba, Adam: A method for stochastic optimization, in: 3rd International Conference on Learning Representations (ICLR 2015), San Diego, USA, 2015. https://arxiv.org/abs/1412.6980 (accessed October 1, 2020).
[73] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in: V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (Eds.), Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science, Springer, Cham, 2018, pp. 833–851, https://doi.org/10.1007/978-3-030-01234-2_49.
[74] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, Focal loss for dense object detection, IEEE Trans. Pattern Anal. Mach. Intell. 42 (2) (2020) 318–327, https://doi.org/10.1109/TPAMI.2018.2858826.
[75] A. Zhang, K.C.P. Wang, B. Li, E. Yang, X. Dai, Y. Peng, Y. Fei, Y. Liu, J.Q. Li, C. Chen, Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network, Comput. Aided Civil Infrastruct. Eng. 32 (10) (2017) 805–819, https://doi.org/10.1111/mice.12297.
[76] S. Chambon, J.-M. Moliard, Automatic road pavement assessment with image processing: review and comparison, Int. J. Geophys. 2011 (2011) 1–20, https://doi.org/10.1155/2011/989354.
[77] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, S. Wang, DeepCrack: learning hierarchical convolutional features for crack detection, IEEE Trans. Image Process. 28 (3) (2019) 1498–1512, https://doi.org/10.1109/TIP.2018.2878966.
[78] Y. Shi, L. Cui, Z. Qi, F. Meng, Z. Chen, Automatic road crack detection using random structured forests, IEEE Trans. Intell. Transp. Syst. 17 (12) (2016) 3434–3445, https://doi.org/10.1109/TITS.2016.2552248.
[79] How transferable are features in deep neural networks? in: J.A.F. Thompson, M. Schonwiesner, Y. Bengio, D. Willett, Z. Ghahramani, M. Welling, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 27 Curran Associates, Inc., Montreal, Canada, 2019, pp. 3320–3328. http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf (accessed October 1, 2020).