Journal of Building Engineering 76 (2023) 107105

Contents lists available at ScienceDirect

Journal of Building Engineering

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

Full length article

# Deep learning for crack detection on masonry façades using limited data and transfer learning

Stamos Katsigiannis [a,*], Saleh Seyedzadeh [b], Andrew Agapiou [c], Naeem Ramzan [d]

[a] Durham University, UK
[b] University of Edinburgh, UK
[c] University of Strathclyde, UK
[d] University of the West of Scotland, UK

ARTICLE INFO

*Keywords:*
Crack detection
Brickwork masonry
Deep learning
Convolutional neural networks
Transfer learning
Brickwork dataset

ABSTRACT

Crack detection in masonry façades is a crucial task for ensuring the safety and longevity of buildings. However, traditional methods are often time-consuming, expensive, and labour-intensive. In recent years, deep learning techniques have been applied to detect cracks in masonry images, but these models often require large amounts of annotated data to achieve high accuracy, which can be difficult to obtain. In this article, we propose a deep learning approach for crack detection on brickwork masonry façades using transfer learning with limited annotated data. Our approach uses a pre-trained deep convolutional neural network model as a feature extractor, which is then optimised specifically for crack detection. To evaluate the effectiveness of our proposed method, we created and curated a dataset of 700 brickwork masonry façade images, and used 500 images for training, 100 for validation, and the remaining 100 images for testing. Results showed that our approach is very effective in detecting cracks, achieving an accuracy and F1-score of up to 100% when following end-to-end training of the neural network, thus being a promising solution for building inspection and maintenance, particularly in situations where annotated data is limited. Moreover, the transfer learning approach can be easily adapted to different types of masonry façades, making it a versatile tool for building inspection and maintenance.

## 1. Introduction

A large number of historical and other existing buildings include brickwork masonry in façades, or in the internal walls in buildings with concrete structure. Cracks in brick masonry are a common problem that can have significant structural and aesthetic implications, being the main issue with the walls that makes structures prone to water damage and mould [1,2]. Non-destructive techniques (NDT), including acoustic emission [3], thermographic inspection [4] and visual inspection by trained personnel [5] have been commonly used for inspection of brick masonry buildings. Manual inspection is typically carried out from the ground, which hinders accruing consistent inspection reports from different personnel [6]. Photogrammetry has also been a solution to help inspectors in making better decisions, as photogrammetry algorithms can be used to transform ground or aerial images to digital

* Corresponding author.
*E-mail address:* stamos.katsigiannis@durham.ac.uk (S. Katsigiannis).

https://doi.org/10.1016/j.jobe.2023.107105
Received 24 March 2023; Received in revised form 8 June 2023; Accepted 13 June 2023
Available online 23 June 2023

maps and 3D models [7]. However, capturing photos from the ground can result in issues such as lower quality of images for higher building levels and difficulties in the localisation of the captured images.

The emergence of commercially available "low-cost" unmanned aerial vehicles (UAVs) offered a solution to these issues by allowing the acquisition of high-quality images and structured point cloud data. Nevertheless, despite having easier access to higher quality images for inspecting brickwork masonry, manual inspection of massive image collections is labour intensive and prone to human errors. Hence, machine vision, specifically machine learning-based image processing, has been deployed for the identification of damages in structures [8]. The majority of the recent works on crack detection has been focused on concrete surfaces and pavements [8,9]. However, due to the lack of an appropriate image dataset, machine learning techniques have not been deployed to identify cracks in brickwork façades.

In this work, we address this research gap and data unavailability by creating and curating a new image dataset for brickwork crack detection and by training and validating various deep learning models for the task of classifying brickwork images as cracked or normal. To this end, images from external brickwork masonry were acquired from a historical building and were combined with images acquired from various online sources. Then, transfer learning was used in order to fine tune widely used pre-trained convolutional neural network (CNN) models for the task at hand, thus further addressing the issue of limited data availability. Experimental results demonstrated the suitability of the proposed approach, achieving a maximum classification accuracy of 100% depending on the base CNN model used and the training strategy.

The main contributions of this work can be summarised as follows: (i) the curation and public release[1] of an image dataset for crack detection on brickwork masonry façades, (ii) a comparative study of various widely used pre-trained CNN models that were fine-tuned for detecting cracks in masonry façades using transfer learning, (iii) an evaluation of various training strategies, including end-to-end training vs. fine tuning only the final classification layers, and using data augmentation techniques to increase the number of training samples vs. using only the original images for training, and (iv) a detailed experimental evaluation and ablation study of the examined CNN-based crack detection models.

## 2. Background

Crack detection in construction is a critical issue that has received significant attention from researchers in recent years [10,11]. The objective of crack detection is to identify and locate cracks in structures and infrastructure, which can lead to the failure of the structure if not addressed. The research in this field has focused on developing new and innovative methods for detecting cracks in a fast, accurate, and non-destructive manner.

Early research on utilising computer vision for monitoring purposes focused on crack detection and was based on heuristic algorithms. Abdel-Qader et al. [12] proposed an edge-detection method for crack identification in bridges comparing different algorithms, including fast Haar transform, fast Fourier transform, Sobel, and Canny edge detectors. This technique applies different filters to extract the edge figure from the façade images. Yu et al. [13] proposed a semi-automatic method based on graph-based search and utilised Sobel edge detection to extract crack properties. Li et al. [14] utilised a wavelet-based algorithm to remove images' noise and a Chan–Vese model for crack segmentation. Nishikawa et al. [15] improved crack segmentation using a genetic algorithm to filter out the images' noise and unwanted subjects, whereas a morphology-based image detector for crack detection was proposed to inspect buried sewers [16]. Later, Sinha et al. [17] used a two-step method using statistical filters to extract crack features locally and identify crack segments using cleaning and linking. Several heuristic methods were also proposed for the identification of other damages. These works include detecting spalling using segmentation [18], steel cracks with region localisation [19], and steel corrosion using the wavelet-based algorithm [20].

The majority of recent automated damage detection techniques involve a machine learning method, particularly an artificial neural network (ANN) [21], to train a model using historical data. Deep learning algorithms such as convolutional neural networks (CNN) for classification, object detection, and segmentation have noticeably enhanced image-based damage identification accuracy. Although these techniques require a sufficiently large dataset to create a reliable prediction model, they provide a much better automation level. Few studies focusing on building inspection utilised techniques other than CNN. Chang et al. [22] applied feed-forward NN to find damage on high rise buildings, localise it, and estimate the intensity for further analysis. Jahanshahi and Masri [23] used a support vector machine for concrete crack detection with measurements based on morphological features.

Kim et al. [24] presented a framework for cracks and other patterns classification using CNN architectures, namely AlexNet for cracks and GoogleNet for spalling. Atha and Jahanshahi [25] also compared different deep architectures for corrosion detection. Cha et al. [26] used CNN object detection to locate the concrete cracks using bounding boxes. The model was trained on low-resolution images and tested over high-resolution images captured by a handheld camera. Yeum et al. [27] proposed a region-based CNN (R-CNN) for post-event building assessment and used a huge structural image dataset to train classification and object detection models. Cha et al. [28] investigated the use of a faster R-CNN to recognise multiple damages (i.e. cracks and different levels of corrosion and delamination) and proposed a technique to localise damages. Zhang et al. [29] introduced CrackNet, a deep CNN architecture for the semantic segmentation of cracks. In crack semantic segmentation, each pixel of the acquired image is classified into the crack or non-crack classes resulting in the identification of the damage's shape. Dung and Anh [30] further explored crack

[1] Dataset download link: https://doi.org/10.5281/zenodo.8014150.

semantic segmentation to determine path and density, whereas Mei and Gul [31] proposed a generative adversarial network and connectivity map to extract pavement cracks at the pixel level of images acquired using a GoPro camera.

Several works developed open datasets for the application of damage detection using machine learning techniques regarding building inspection. Maguire et al. [32] created a concrete crack image dataset that included 56,000 images classified as crack or non-crack, whereas Xu et al. [33] developed a dataset of 6069 bridge crack images. Although there has been much research work on using ML-based damage detection, most works have used limited data for training the models and have not provided public access to the generated datasets.

The literature suggests that deep learning-based methods have demonstrated good results in crack detection on masonry façades. However, the performance of these models largely depends on the availability of training data. Limited availability of annotated data is a common problem in many real-world applications, including crack detection on masonry façades. When a reasonably large dataset is not available, it is possible to fine-tune a pre-trained network with a smaller image dataset to train an accurate model [34], an approach called "transfer learning". Transfer learning allows the use of pre-trained models on related tasks to improve the performance of the task at hand, e.g. fine-tuning a model pre-trained for classifying natural images for detecting cracks in masonry. This approach has been applied to several tasks in computer vision, including image classification, object detection, and semantic segmentation [35].

In recent years, transfer learning has been used for the inspection of building façades, with the goal of detecting anomalies in the brickwork [36]. In addition, Gopalakrishan et al. [37] deployed a pre-trained network to develop a classifier for identifying cracks in asphalt and concrete surfaces, whereas Zhang et al. [38] developed a framework for adopting transfer learning and CNN for classifying pavement images into the "crack", "repaired crack", and "background" classes. Dais et al. [39] used deep learning to train a model for crack detection on images from masonry walls, achieving a 95.3% accuracy and on pixel level with 79.6% F1-score.

## 3. Material and methods

To address the problem of automated detection of cracks in brickwork masonry, we first created and curated a dataset containing images of brickwork with and without cracks. Then, we examined the use of deep learning approaches for identifying "crack" and "no-crack" brickwork images by employing pre-trained convolutional neural networks that were fine-tuned on our brickwork dataset.

### *3.1. The Brickwork Cracks Dataset*

The proposed *Brickwork Cracks Dataset* was created in order to facilitate the use of transfer learning for the creation and training of machine learning models for the task of classifying brickwork images as belonging to one of the "crack" or "non-crack" classes. One of the key challenges in using transfer learning for brickwork crack detection is the creation of a suitable dataset. A large and diverse dataset is essential to train deep learning models effectively, and this is particularly important in transfer learning, where the pre-trained model must be adapted to the task at hand. To this end, two sources were used to create the proposed dataset:

(a) A database of images taken from the external façade of the Architecture School of the University of Strathclyde, Glasgow, that includes various images of the brick masonry walls of regular pattern (ignoring rubble masonry). The images were acquired during the development of a smart mobile application for "Building Façade Defect Inspection" based on the integration of methodologies and tools, including virtual reality, digital photogrammetry and mobile app technologies to collect real-time data that support automated decision making [40]. The database included masonry walls with cracks, with windows and doors, with varied in colour masonry units, as well as with varied illumination and capture-angle.

(b) Various online sources, including online image databases, brick manufacturer websites, and academic publications. The images were screened for quality, and those that were deemed suitable for the task were included in the proposed dataset.

The images were then annotated to indicate the presence of cracks in the brickwork. This was achieved using a custom-built annotation tool, which allowed the annotators to draw bounding boxes around the cracks and label them as either horizontal, vertical, or diagonal. The manual labelling process was time-consuming and required a high level of expertise, making it a challenge to scale the process to larger datasets. Labelling is a critical step in the process because it helps the machine learning algorithm learn what to look for and what to ignore. In addition to the bounding boxes referring to cracks in the brickwork, bounding boxes for brickwork without damage were also annotated by the experts. The image regions denoted by the annotated bounding boxes were stored as separate images and rescaled to $227 \times 227$ pixels for consistency. This process led to 350 images for the "crack" class and 350 images for the "non-crack" class, for a total of 700 images for the proposed dataset. The dataset is perfectly balanced across the two classes (50% "crack" - 50% "non-crack"), making it ideal for the training of machine learning models. Some examples of images from the proposed dataset are shown in Fig. 1.

Furthermore, in order to provide a fair performance evaluation and avoid over-fitting the machine learning models during training, stratified sampling was used to randomly divide the dataset into a training, a validation, and a test set, containing 500 (approx. 71.4%), 100 (approx. 14.3%), and 100 (approx. 14.3%) images respectively, with all the sets having a perfect balance between the "crack" (50%) and "non-crack" (50%) classes. All the examined machine learning models in this work were trained using images only from the training set, with the validation set used to select the best performing models, and the "unseen" test set only used to report the final performance of the models.

The proposed dataset will be released publicly [41] in order to guarantee the replicability of our study and to facilitate further research in the field.

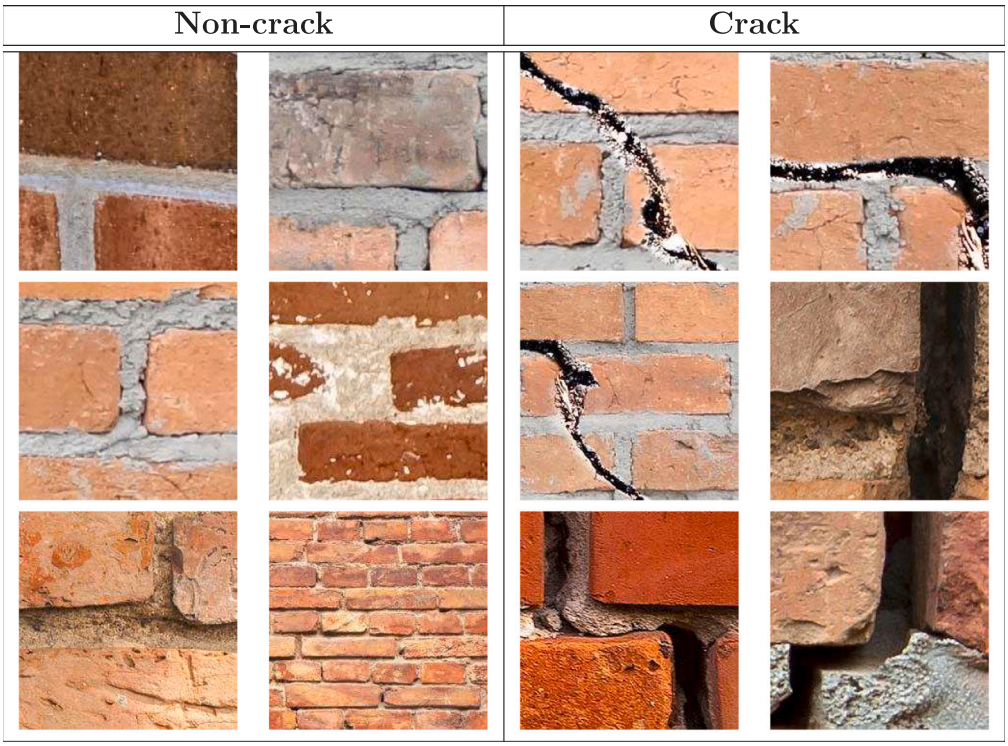


**Fig. 1.** Sample images from the proposed dataset for the "non-crack" (left) and the "crack" (right) classes.

### 3.2. Data augmentation

The lack of sufficiently large datasets has been one of the main obstacles in applying deep learning approaches in various image classification problems. Depending on the application, the acquisition of additional data can be arduous and expensive, in terms of time and required resources, or even outright impossible due to the rarity of the conditions needed for acquiring them. To address the issue of data scarcity in image classification applications where few data samples are available, data augmentation has been proposed and proven to be very effective [42]. Data augmentation is the process of artificially creating new data out of the existing ones by applying various operations on them. It has been widely used by researchers in image processing as it can improve the performance of deep learning models, assist in avoiding over-fitting, and allow the expansion of limited datasets in order to exploit the capabilities and benefits of "big data" [43].

Given the limited amount of images (700) in our brickwork cracks dataset, we opted to also examine the use of data augmentation in this work in order to increase the amount of available data, thus allowing the use of more complex deep learning image classification models, while avoiding over-fitting. To this end, each image in the dataset was used as the source for additional images created at each iteration of the training procedure, by randomly flipping them horizontally, randomly flipping them vertically, randomly shifting the brightness between 0.3 and 1.0, and randomly rotating them between $0°$ and $45°$. It must also be noted that the pixel values of all images were rescaled by a factor of $1/255$ before being used with the machine learning models and that all the experiments in this work were conducted with and without data augmentation in order to compare the respective performance. The Keras [44] *ImageDataGenerator* class was used for the creation of the augmented images and all random values for the augmentation operations were generated using a uniform probability distribution. Furthermore, it must also be highlighted that the creation and use of additional (augmented) images was performed for training the machine learning models, using only the images in the training set as the source. Consequently, any reported results on the test data refer to original images only.

### 3.3. Classification using deep convolutional neural networks

Given the relatively small number of images in the proposed dataset, transfer learning was selected as the optimal strategy for the training of machine learning models for the task of distinguishing images as belonging to the "crack" or "non-crack" classes. To this end, our approach is based on well-established deep learning architectures that have been trained using a sufficiently large image dataset and have been shown to be efficient feature extractors for image classification tasks. We selected six CNN architectures that have been previously trained on the ImageNet [45] dataset, which contains 1000 image classes, spread across 1.4 million

**Table 1**
Number of parameters (in millions) of the examined models.

| Base model | Parameters |
|---|---|
| VGG16 | 134.27M |
| VGG19 | 139.58M |
| MobileNetV2 | 2.26M |
| InceptionResNetV2 | 54.34M |
| InceptionV3 | 21.81M |
| Xception | 20.87M |

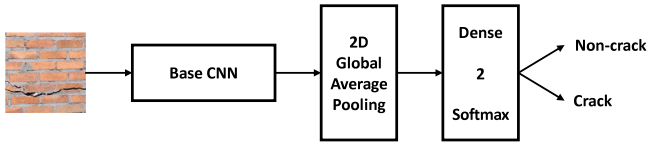


**Fig. 2.** Overview of the proposed method. This approach was followed for all the examined CNN models, except for VGG16 and VGG19 that use a flatten layer and three dense layers after the pre-trained convolutional base, two of size 4096 using a ReLU activation function, followed by one of size 2. The architecture of the base CNN differs according to the architecture of the selected pre-trained model.

images. The selected architectures were VGG16 [46], VGG19 [46], MobileNetV2 [47], InceptionResNetV2 [48], InceptionV3 [49], and Xception [50]. To fine-tune the pre-trained models for classifying between brickwork images with cracks and without cracks, the output of the pre-trained convolutional base of the models was first passed to a 2D global average pooling layer, followed by a fully-connected (dense) output layer with 2 neurons, as many as the number of classes in our task. Furthermore, a softmax activation function was used in the output layer. It must be noted that the VGG16 and VGG19 models use a flatten layer and three fully-connected layers in their output [46], two of size 4096 using a ReLU activation function, followed by one of size 2 that uses a softmax activation function. Despite the other architectures using a single fully-connected layer after the convolutional base, we opted to follow the original design for VGG16 and VGG19 to ensure a fair experimental comparison. An overview of the described neural network architecture is provided in Fig. 2, whereas an overview of the total number of parameters for each model is provided in Table 1.

To train the models, we used Cross-Entropy as the loss function, defined as:

$$L_{CE} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{1}$$

where $c$ is the class, $M$ is the number of classes (2 in this work), $y_{o,c}$'s value is 1 if observation $o$ belongs to class $c$, otherwise it is 0, and $p_{o,c}$ is the predicted probability that $o$ belongs to class $c$. Furthermore, a batch size of 32 was selected for the training and the Adam optimiser was used with a learning rate of $10^{-5}$, whereas training for each model stopped after 20 epochs with no improvement in validation accuracy. All experiments were implemented using the Python programming language and the Keras and Tensorflow frameworks [51]. Four different strategies were examined and evaluated for the training of the proposed models:

1. Models were trained end-to-end and data augmentation was used to increase the number of training images.
2. Models were trained end-to-end without any data augmentation.
3. The weights of the convolutional base of the models were frozen and only the additional fully-connected layers were trained. Data augmentation was used to increase the number of training images.
4. The weights of the convolutional base of the models were frozen and only the additional fully-connected layers were trained. No data augmentation used.

It must be noted that to freeze the weights of the convolutional base of the models in the respective strategies, the "trainable" parameter of the Keras implementation of the models was set to "false" for all the layers included in the convolutional base. This led to the weights of the respective layers not being updated during back-propagation.

During end-to-end training, the pre-trained convolutional base of the network is fine-tuned to the downstream task, in addition to training the final classification layer(s). As a result, the fine-tuned convolutional base becomes an image feature extractor that is optimised for distinguishing between brickwork images with and without cracks. Contrary to end-to-end training, when the convolutional base's weights are frozen, then the convolutional base acts as a generic image feature extractor, having the benefit of not requiring any additional training, thus leading to lower training time as only the final classification layer(s) needs to be trained.

## 4. Results

The performance of the six deep convolutional neural network architectures for the four examined training strategies was evaluated by performing supervised classification experiments on the proposed dataset. All models were trained on the training

**Table 2**
Classification results for end-to-end training using data augmentation.

| Base model | Accuracy | F1-score | Precision | Recall | Jaccard |
|---|---|---|---|---|---|
| VGG16 | 0.9700 | 0.9700 | 0.9702 | 0.9700 | 0.9417 |
| VGG19 | 0.9900 | 0.9900 | 0.9902 | 0.9900 | 0.9802 |
| MobileNetV2 | **1** | **1** | **1** | **1** | **1** |
| InceptionResNetV2 | **1** | **1** | **1** | **1** | **1** |
| InceptionV3 | 0.9900 | 0.9900 | 0.9902 | 0.9900 | 0.9802 |
| Xception | 0.9700 | 0.9700 | 0.9717 | 0.9700 | 0.9417 |

**Table 3**
Classification results for end-to-end training without using data augmentation.

| Base model | Accuracy | F1-score | Precision | Recall | Jaccard |
|---|---|---|---|---|---|
| VGG16 | 0.9900 | 0.9900 | 0.9902 | 0.9900 | 0.9802 |
| VGG19 | 0.9900 | 0.9900 | 0.9902 | 0.9900 | 0.9802 |
| MobileNetV2 | 0.9600 | 0.9600 | 0.9607 | 0.9600 | 0.923 |
| InceptionResNetV2 | 0.9800 | 0.9800 | 0.9808 | 0.9800 | 0.9608 |
| InceptionV3 | 0.9900 | 0.9900 | 0.9902 | 0.9900 | 0.9802 |
| Xception | **1** | **1** | **1** | **1** | **1** |

**Table 4**
Classification results when the weights of the convolutional base are kept frozen and data augmentation is used.

| Base model | Accuracy | F1-score | Precision | Recall | Jaccard |
|---|---|---|---|---|---|
| VGG16 | **0.9200** | **0.9199** | **0.9227** | **0.9200** | **0.8516** |
| VGG19 | 0.8500 | 0.8493 | 0.857 | 0.8500 | 0.7382 |
| MobileNetV2 | 0.6800 | 0.6716 | 0.7005 | 0.6800 | 0.5079 |
| InceptionResNetV2 | 0.8300 | 0.8298 | 0.8312 | 0.8300 | 0.7092 |
| InceptionV3 | 0.7600 | 0.7478 | 0.8224 | 0.7600 | 0.6003 |
| Xception | 0.7400 | 0.7292 | 0.7857 | 0.7400 | 0.5766 |

**Table 5**
Classification results when the weights of the convolutional base are kept frozen and data augmentation is not used.

| Base model | Accuracy | F1-score | Precision | Recall | Jaccard |
|---|---|---|---|---|---|
| VGG16 | 0.9200 | 0.9200 | 0.9200 | 0.9200 | 0.8519 |
| VGG19 | **0.9300** | **0.9299** | **0.9316** | **0.9300** | **0.8691** |
| MobileNetV2 | 0.8100 | 0.8098 | 0.8111 | 0.8100 | 0.6805 |
| InceptionResNetV2 | 0.8100 | 0.8095 | 0.8131 | 0.8100 | 0.6801 |
| InceptionV3 | 0.8400 | 0.8394 | 0.8450 | 0.8400 | 0.7234 |
| Xception | 0.7900 | 0.7852 | 0.8187 | 0.7900 | 0.6475 |

dataset, and at each epoch, performance was measured for the validation set. The model from the epoch that provided the best performance on the validation set was then selected as the best model for each experiment and its performance was then evaluated on the unseen test set. Consequently, all the performance metrics reported in this section refer to the performance on the unseen test set, in order to provide a fair experimental comparison. The computed performance metrics were the classification accuracy, F1-score, precision, recall, and Jaccard index [52]. Furthermore, since the F1-score, precision, and recall depend on which class is considered as positive, their reported scores in this work are the average scores between the two examined classes ("crack" and "non-crack").

Results for the four training strategies using all the examined models are reported in Tables 2 to 5 in terms of the accuracy, F1-score, precision, recall, and Jaccard index metrics, while confusion matrices for the test set are provided in Figs. 3 to 6. The maximum classification performance reached 100% in terms of all the examined metrics for the end-to-end training strategy using the MobileNetV2 [47] and InceptionResNetV2 [48] models with data augmentation, and the Xception [50] model without data augmentation. The strategy of freezing the weights of the convolutional base of the models underperformed significantly, achieving a maximum F1-score of 91.99% using the VGG16 model with data augmentation, and a maximum F1-score of 92.99% with the VGG19 model without data augmentation. Based on these results, it is evident that the end-to-end training strategy provides superior performance for the task of detecting cracks in brickwork images using the examined CNN-based models.

Table 2 reports results for end-to-end training using data augmentation. Both MobileNetV2 [47] and InceptionResNetV2 [48] achieved perfect classification results, having an accuracy and F1-score of 100%, while the F1-scores for the other compared models ranged between 97%–99%. Results for the end-to-end strategy without using data augmentation are reported in Table 3. In this case, the Xception [50] model achieved perfect classification performance, having an accuracy and F1-score of 100%, while the F1-scores for the other compared models ranged between 96%–99%. Performance was significantly worse when the weights of the convolutional base were frozen (not fine-tuned) and only the additional fully-connected layers were trained. In the case of the

**VGG16**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 49 | 1 |
| Crack | 2 | 48 |

**VGG19**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 49 | 1 |
| Crack | 0 | 50 |

**MobileNetV2**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 50 | 0 |
| Crack | 0 | 50 |

**InceptionResNetV2**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 50 | 0 |
| Crack | 0 | 50 |

**InceptionV3**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 49 | 1 |
| Crack | 0 | 50 |

**Xception**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 47 | 3 |
| Crack | 0 | 50 |

**Fig. 3.** Confusion matrices for end-to-end training using data augmentation.

**VGG16**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 50 | 0 |
| Crack | 1 | 49 |

**VGG19**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 50 | 0 |
| Crack | 1 | 49 |

**MobileNetV2**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 49 | 1 |
| Crack | 3 | 47 |

**InceptionResNetV2**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 48 | 2 |
| Crack | 0 | 50 |

**InceptionV3**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 49 | 1 |
| Crack | 0 | 50 |

**Xception**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 50 | 0 |
| Crack | 0 | 50 |

**Fig. 4.** Confusion matrices for end-to-end training without using data augmentation.

**VGG16**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 44 | 6 |
| Crack | 2 | 48 |

**VGG19**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 39 | 11 |
| Crack | 4 | 46 |

**MobileNetV2**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 26 | 24 |
| Crack | 8 | 42 |

**InceptionResNetV2**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 43 | 7 |
| Crack | 10 | 40 |

**InceptionV3**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 27 | 23 |
| Crack | 1 | 49 |

**Xception**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 27 | 23 |
| Crack | 3 | 47 |

**Fig. 5.** Confusion matrices for when the weights of the convolutional base are kept frozen and data augmentation is used.

**VGG16**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 46 | 4 |
| Crack | 4 | 46 |

**VGG19**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 48 | 2 |
| Crack | 5 | 45 |

**MobileNetV2**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 39 | 11 |
| Crack | 8 | 42 |

**InceptionResNetV2**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 38 | 12 |
| Crack | 7 | 43 |

**InceptionV3**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 39 | 11 |
| Crack | 5 | 45 |

**Xception**

| Actual \ Predicted | Non-crack | Crack |
|---|---|---|
| Non-crack | 32 | 18 |
| Crack | 3 | 47 |

**Fig. 6.** Confusion matrices for when the weights of the convolutional base are kept frozen and data augmentation is not used.

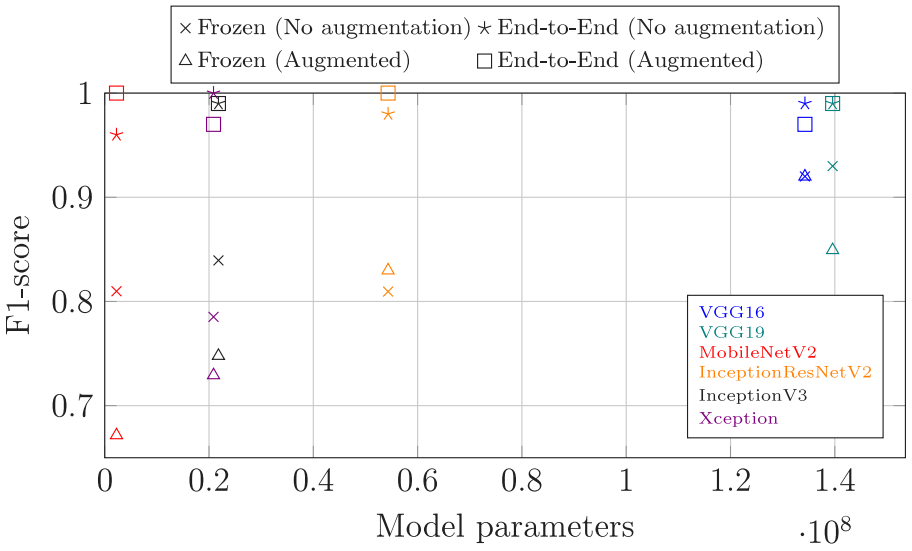


**Fig. 7.** F1-score achieved by each model versus its number of parameters. MobileNetV2 using end-to-end training and data augmentation (shown in red) achieved the best performance (F1-score = 100%) for the lowest number of parameters.

frozen convolutional base with data augmentation, VGG16 [46] achieved the best performance, reaching an accuracy of 92% and an F1-score of 91.99%, whereas the other models performed considerably worse, achieving F1-scores ranging between 67.16%–84.93%, as shown in Table 4. Performance was marginally better in the case of frozen convolutional base without data augmentation, with VGG19 [46] and VGG16 [46] achieving accuracies of 93% and 92%, and F1-scores of 92.99% and 92%, respectively, whereas the F1-scores for the other models ranged between 78.52%–83.94%, as shown in Table 5.

## 5. Discussion

As shown in Section 4, our experimental results were inconclusive regarding to which model is the most suitable for the task of cracks detection in brickwork images, as all of the MobileNetV2 [47] (with data augmentation), InceptionResNetV2 [48] (with data augmentation), and Xception [50] (without data augmentation) models achieved a perfect classification performance when using end-to-end training. Nevertheless, as can be seen in Table 1, these three models differ significantly in terms of their size, with the MobileNetV2-based model having 2.26 million parameters, the Xception-based model 20.87 million parameters, and the InceptionResNetV2-based model 54.34 million parameters. In addition, many of the examined models across the different training strategies achieved similar performance among each other. However it is evident from Fig. 7 that equally good classification performance can be achieved using models with less parameters. Training plots for the best performing models are provided in Fig. 8 for the end-to-end training strategy, and in Fig. 9 for the strategy of freezing the weights of the convolutional base. It must be noted that the number of training epochs varies across the models, as training was stopped after 20 consecutive epochs without any improvement in the validation accuracy.

Considering the computational cost for performing the inference using these models and the desired ability to deploy the developed model on handheld devices (e.g. mobile phones, tablets) for conducting inspections on brickwork, it is evident that the MobileNetV2-based model, created using end-to-end training with data augmentation, can be considered as the most appropriate model for the task at hand, as it can achieve the best performance, and its size makes it suitable for devices with low computational capabilities. It can be argued however that the smaller size of the model may affect negatively its ability to generalise to new brickwork images. For our experimental evaluation, all results were reported for a completely “unseen” subset of the proposed dataset, in order to ensure a fair performance comparison that would provide a good estimate of the ability of each model to generalise to new input images.

Despite their popularity, deep learning-based image classification models have long been considered as “black-box” models due to their lack of interpretability or explainability, i.e. the ability to explain the reasons that led a trained machine learning model to a specific prediction [53]. This lack of explainability has been shown to be a significant obstacle to the real-world adoption of deep learning models for image classification within various fields. The inability to explain or justify a trained model’s predictions constitutes a serious obstacle to establishing trust to the model, especially in applications and professions were a chain of liability must be established. To this end, we evaluated the interpretability of the MobileNetV2-based model by using the Gradient-weighted Class Activation Mapping (Grad-CAM) [54] method, which is a localisation technique that is used to generate class-related visual explanations from a trained CNN-based network. Grad-CAM was used in order to examine the regions of the brickwork images

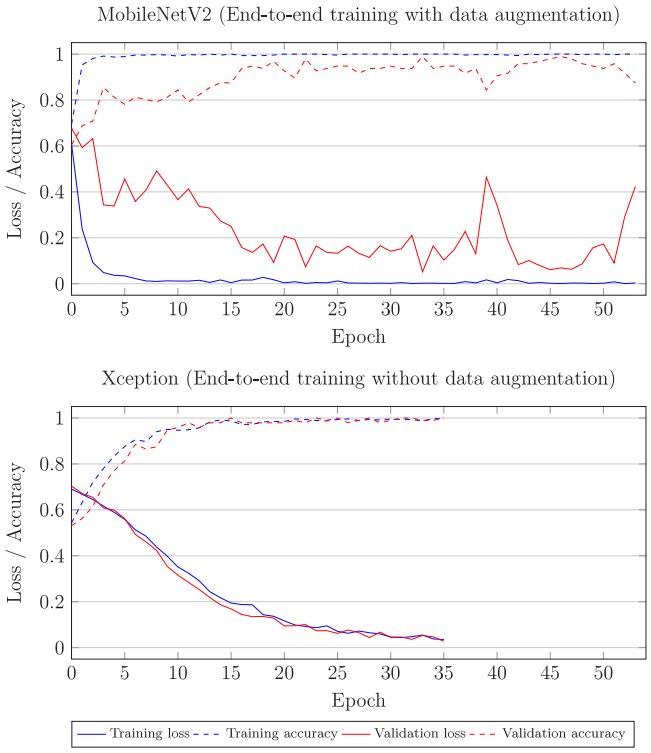


**Fig. 8.** Training plots for the best performing models using the end-to-end training strategy, with data augmentation (MobileNetV2) and without data augmentation (Xception).

that are taken into consideration by the MobileNetV2-based model in order to assign an image to the "crack" class. To this end, Grad-CAM heat maps depicting the class activation for the "crack" class using the last convolutional layer of the network were created for the images in the examined dataset. Fig. 10 depicts the Grad-CAM heat maps for 12 images of the "crack" class, overlaid on the original images. From these images, it is evident that the MobileNetV2-based model indeed relies on the regions of the images that contain cracks in the brickwork in order to make its prediction, indicating that it can be reliably used for the task at hand. The real-world deployment of the proposed methodology for crack detection in brickwork masonry could benefit significantly by the inclusion of such heat maps in the output of the used models. Building professionals would be able to visually verify the used model's predictions by inspecting the produced heat maps and cross-referencing them to the model's predictions, thus enabling them to build trust towards the system.

Regarding the examined training strategies, it is evident from Tables 2, 3, 4, and 5 that the end-to-end training strategy performs significantly better compared to keeping the weights of the convolutional base frozen and training only the final fully-connected layers. The brickwork images used in this work exhibit considerable differences in terms of texture, structure, and colour, compared to the generic images contained in the ImageNet [45,55] dataset that was used to pre-train the examined CNN models. As a result, despite the convolutional base of the pre-trained models being very efficient feature extractors for generic images, fine-tuning is required to adapt the computed features to the downstream task, i.e. classifying brickwork images as cracked or not cracked.

The field of image classification using deep neural networks has advanced significantly in recent years. Various methods that outperform CNNs on standard image classification benchmarks, such as ImageNet [45,55], CIFAR [56], JFT [57], and others, have been recently proposed and are mainly based on the transformer [58] architecture. Architectures like the Vision Transformer (ViT) [59], the Image Enhanced Vision Transformer (IEViT) [60], the Swin Transformer [61], and others [62–65] have demonstrated impressive performance in image classification and object detection tasks, but come at a cost of increased computational complexity and the requirement for considerably large training datasets in order to train well-performing models. In this work we opted to not evaluate the performance of transformer-based models given that there was no room for improving the best classification performance achieved using the examined CNN models (accuracy and F1-score of 100%). Furthermore, the increased computational complexity of transformer-based models would complicate the deployment of the trained models on handheld devices with low computational capabilities, while the small size of the available brickwork dataset would potentially hinder our ability to train them efficiently.

Considering our experimental results, it is evident that the proposed approach achieved impressive performance for the task of crack detection in brickwork masonry. We hypothesise that the reason for such performance lies in the use of a sufficiently diverse dataset that included brickwork masonry that varied in colour, illumination, and capture angle; the focus on a very specific

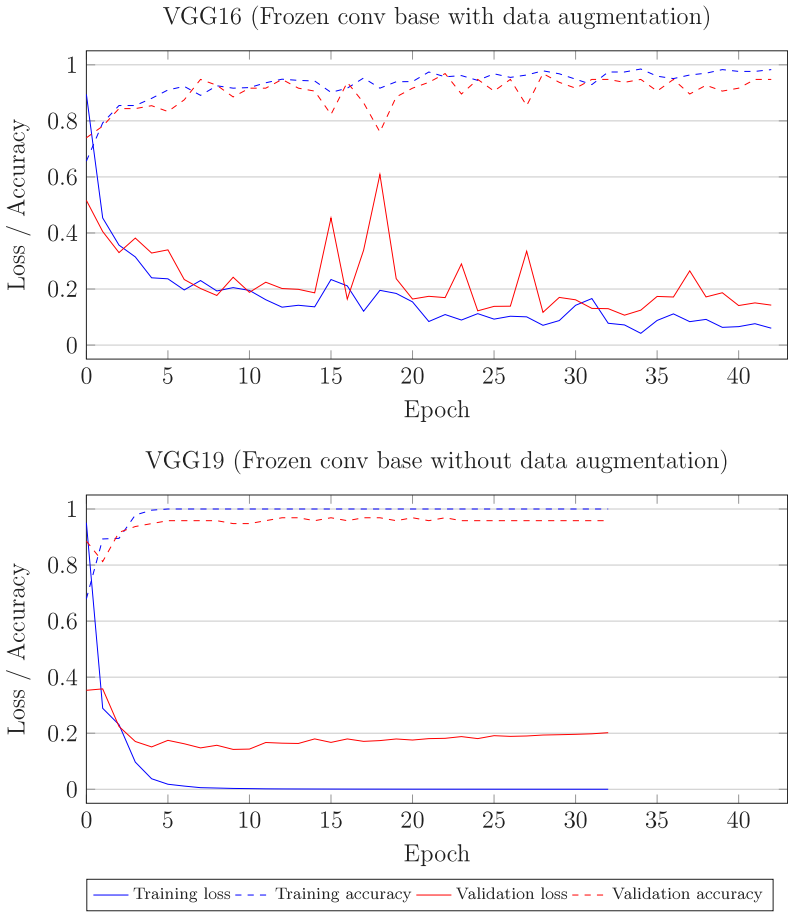


**Fig. 9.** Training plots for the best performing models using the strategy of freezing the weights of the convolutional base, with data augmentation (VGG16) and without data augmentation (VGG19).

brickwork deformation, i.e. cracks, ignoring other potential deformations, e.g. spalling; and focusing only on detecting cracks without examining their severity, thus simplifying the problem.

## 6. Conclusion

The objective of this study was to propose and evaluate a deep learning approach for automating the detection of cracks in brickwork masonry using imaging techniques. The main challenge was the unavailability of suitable data, which we addressed by creating and curating a comprehensive dataset consisting of brickwork images with and without cracks. This dataset has been made publicly available to facilitate further research in this field.

To develop our approach, we used transfer learning to fine-tune popular pre-trained convolutional neural networks. We trained the models on the curated dataset and performed supervised classification experiments. Results showed that the proposed approach can efficiently detect cracks in brickwork masonry, achieving a 100% accuracy and F1-score using the MobileNetV2, InceptionresNetV2, and Xception-based models. This performance was achieved using an end-to-end training strategy and data augmentation for the first two models, whereas no data augmentation was used for the latter. However, the MobileNetV2-based model can be considered as the most suitable for this task due to its small size, making it ideal for use on handheld mobile devices and UAVs. We also compared the performance of end-to-end training with that of training only the final classification layers while keeping the weights of the convolutional base frozen. The former approach consistently outperformed the latter, achieving a maximum F1-score of 100% compared to 92.99%. These findings demonstrate the effectiveness of our proposed approach for detecting cracks in brickwork masonry using deep learning techniques.

Our future work will focus on integrating the proposed models into real-world systems for detecting cracks in brickwork masonry. These systems will utilise camera-equipped handheld devices, such as mobile phones and tablets, as well as commercial UAVs. By deploying these methods in the field, we can evaluate their performance, reliability, and generalisation ability for building

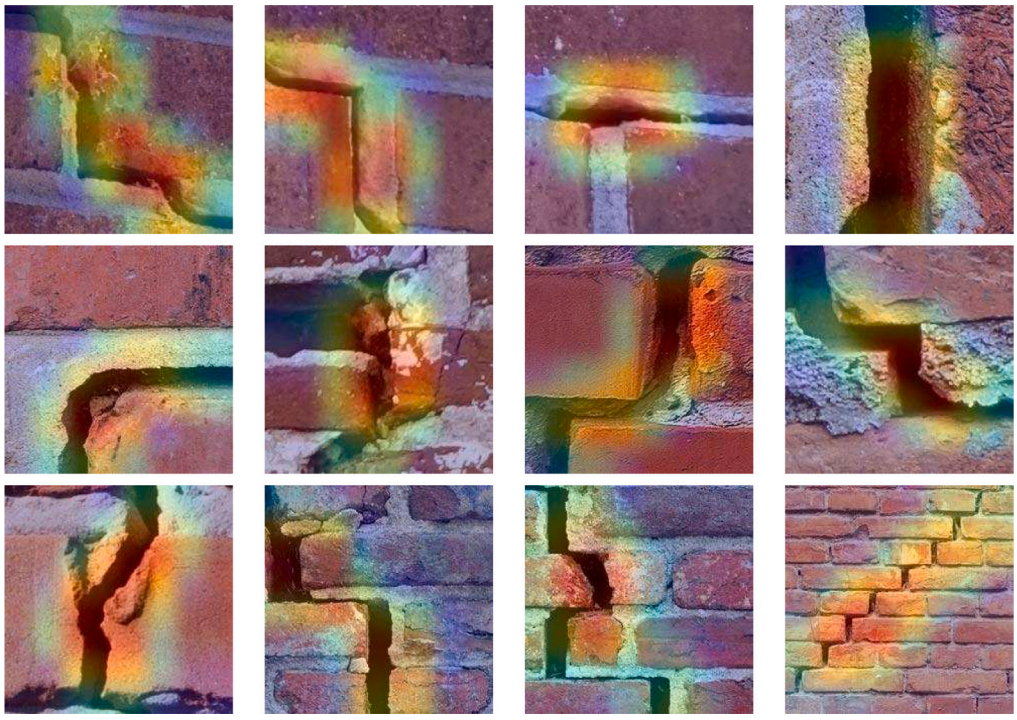

**Fig. 10.** Grad-CAM heat maps, overlaid on the original images for the "crack" class.

inspections. It is important to note that the proposed methods can currently only detect the presence of cracks in brickwork masonry. Therefore, we plan to expand the dataset and explore deep learning approaches to assess the severity of detected cracks. This will help us to improve the accuracy and practicality of our models for real-world applications.

## 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.

## Data availability

A link to download the data has been added to the manuscript.

## Acknowledgments

This research was funded under the Built Environment - Smart Transformation's (formerly the Construction Scotland Innovation Centre) I-CON programme.

## References

[1] J.R. Casas, Reliability-based assessment of masonry arch bridges, Constr. Build. Mater. 25 (4) (2011) 1621–1631, http://dx.doi.org/10.1016/j.conbuildmat.2010.10.011.
[2] A. Soleymani, M.A. Najafgholipour, A. Johari, An experimental study on the mechanical properties of solid clay brick masonry with traditional mortars, J. Build. Eng. 58 (2022) 105057, http://dx.doi.org/10.1016/j.jobe.2022.105057.
[3] K. Ohno, M. Ohtsu, Crack classification in concrete based on acoustic emission, Constr. Build. Mater. 24 (12) (2010) 2339–2346, http://dx.doi.org/10.1016/j.conbuildmat.2010.05.004.
[4] E.Z. Kordatos, D.A. Exarchos, C. Stavrakos, A. Moropoulou, T.E. Matikas, Infrared thermographic inspection of murals and characterization of degradation in historic monuments, Constr. Build. Mater. 48 (2013) 1261–1265, http://dx.doi.org/10.1016/j.conbuildmat.2012.06.062.
[5] Z. Orbán, M. Gutermann, Assessment of masonry arch railway bridges using non-destructive in-situ testing methods, Eng. Struct. 31 (10) (2009) 2287–2298, http://dx.doi.org/10.1016/j.engstruct.2009.04.008.
[6] D.F. Laefer, J. Gannon, E. Deely, Reliability of crack detection methods for baseline condition assessments, J. Infrastr. Syst. 16 (2) (2010) 129–137, http://dx.doi.org/10.1061/(asce)1076-0342(2010)16:2(129).
[7] R. Yu, P. Li, J. Shan, H. Zhu, Structural state estimation of earthquake-damaged building structures by using UAV photogrammetry and point cloud segmentation, Measurement 202 (2022) 111858, http://dx.doi.org/10.1016/j.measurement.2022.111858.

[8] M. Azimi, A.D. Eslamlou, G. Pekcan, Data-driven structural health monitoring and damage detection through deep learning: State-ofthe- art review, Sensors (Switzerland) 20 (10) (2020) 2778, http://dx.doi.org/10.3390/s20102778.
[9] Y.-A. Hsieh, Y.J. Tsai, Machine learning for crack detection: Review and model performance comparison, J. Comput. Civ. Eng. 34 (5) (2020) 04020038, http://dx.doi.org/10.1061/(asce)cp.1943-5487.0000918.
[10] K. Chen, G. Reichard, X. Xu, A. Akanmu, Automated crack segmentation in close-range building façade inspection images using deep learning techniques, J. Build. Eng. 43 (2021) 102913, http://dx.doi.org/10.1016/j.jobe.2021.102913.
[11] Y. Yu, B. Samali, M. Rashidi, M. Mohammadi, T.N. Nguyen, G. Zhang, Vision-based concrete crack detection using a hybrid framework considering noise effect, J. Build. Eng. 61 (2022) 105246, http://dx.doi.org/10.1016/j.jobe.2022.105246.
[12] I. Abdel-Qader, O. Abudayyeh, M.E. Kelly, Analysis of edge-detection techniques for crack identification in bridges, J. Comput. Civ. Eng. 17 (4) (2003) 255–263, http://dx.doi.org/10.1061/(asce)0887-3801(2003)17:4(255).
[13] S.N. Yu, J.H. Jang, C.S. Han, Auto inspection system using a mobile robot for detecting concrete cracks in a tunnel, Autom. Constr. 16 (3) (2007) 255–261, http://dx.doi.org/10.1016/j.autcon.2006.05.003.
[14] G. Li, S. He, Y. Ju, K. Du, Long-distance precision inspection method for bridge cracks with image processing, Autom. Constr. 41 (2014) 83–95, http://dx.doi.org/10.1016/j.autcon.2013.10.021.
[15] T. Nishikawa, J. Yoshida, T. Sugiyama, Y. Fujino, Concrete crack detection by multiple sequential image filtering, Comput.-Aided Civ. Infrastruct. Eng. 27 (1) (2012) 29–47, http://dx.doi.org/10.1111/j.1467-8667.2011.00716.x.
[16] S. Iyer, S.K. Sinha, Segmentation of pipe images for crack detection in buried sewers, Comput.-Aided Civ. Infrastruct. Eng. 21 (6) (2006) 395–410, http://dx.doi.org/10.1111/j.1467-8667.2006.00445.x.
[17] S.K. Sinha, P.W. Fieguth, Automated detection of cracks in buried concrete pipe images, Autom. Constr. 15 (1) (2006) 58–72, http://dx.doi.org/10.1016/j.autcon.2005.02.006.
[18] S.G. Paal, J.-S. Jeon, I. Brilakis, R. DesRoches, Automated damage index estimation of reinforced concrete columns for post-earthquake evaluations, J. Struct. Eng. 141 (9) (2015) 04014228, http://dx.doi.org/10.1061/(asce)st.1943-541x.0001200.
[19] C.M. Yeum, S.J. Dyke, Vision-based automated crack detection for bridge inspection, Comput.-Aided Civ. Infrastruct. Eng. 30 (10) (2015) 759–770, http://dx.doi.org/10.1111/mice.12141.
[20] M.R. Jahanshahi, S.F. Masri, Parametric performance evaluation of wavelet-based corrosion detection algorithms for condition assessment of civil infrastructure systems, J. Comput. Civ. Eng. 27 (4) (2013) 345–357, http://dx.doi.org/10.1061/(asce)cp.1943-5487.0000225.
[21] S. Seyedzadeh, F.P. Rahimian, I. Glesk, M. Roper, Machine learning for estimation of building energy consumption and performance: a review, Vis. Eng. 6 (1) (2018) 5, http://dx.doi.org/10.1186/s40327-018-0064-7.
[22] C.M. Chang, T.K. Lin, C.W. Chang, Applications of neural network models for structural health monitoring based on derived modal properties, Measurement 129 (2018) 457–470, http://dx.doi.org/10.1016/j.measurement.2018.07.051.
[23] M.R. Jahanshahi, S.F. Masri, A new methodology for non-contact accurate crack width measurement through photogrammetry for automated structural safety evaluation, Smart Mater. Struct. 22 (3) (2013) 35019, http://dx.doi.org/10.1088/0964-1726/22/3/035019.
[24] H. Kim, E. Ahn, M. Shin, S.H. Sim, Crack and noncrack classification from concrete surface images using machine learning, Struct. Health Monit. 18 (3) (2019) 725–738, http://dx.doi.org/10.1177/1475921718768747.
[25] D.J. Atha, M.R. Jahanshahi, Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection, Struct. Health Monit. 17 (5) (2018) 1110–1128, http://dx.doi.org/10.1177/1475921717737051.
[26] Y.J. Cha, W. Choi, O. Büyüköztürk, Deep learning-based crack damage detection using convolutional neural networks, Comput.-Aided Civ. Infrastruct. Eng. 32 (5) (2017) 361–378, http://dx.doi.org/10.1111/mice.12263.
[27] C.M. Yeum, S.J. Dyke, J. Ramirez, Visual data classification in post-event building reconnaissance, Eng. Struct. 155 (2018) 16–24, http://dx.doi.org/10.1016/j.engstruct.2017.10.057.
[28] 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 Civ. Infrastruct. Eng. 33 (9) (2018) 731–747, http://dx.doi.org/10.1111/mice.12334.
[29] A. Zhang, K.C. Wang, Y. Fei, Y. Liu, C. Chen, G. Yang, J.Q. Li, E. Yang, S. Qiu, Automated pixel-level pavement crack detection on 3D asphalt surfaces with a recurrent neural network, Comput.-Aided Civ. Infrastruct. Eng. 34 (3) (2019) 213–229, http://dx.doi.org/10.1111/mice.12409.
[30] C.V. Dung, L.D. Anh, Autonomous concrete crack detection using deep fully convolutional neural network, Autom. Constr. 99 (2019) 52–58, http://dx.doi.org/10.1016/j.autcon.2018.11.028.
[31] Q. Mei, M. Gül, A cost effective solution for pavement crack inspection using cameras and deep neural networks, Constr. Build. Mater. 256 (2020) 119397, http://dx.doi.org/10.1016/j.conbuildmat.2020.119397.
[32] M. Maguire, S. Dorafshan, R.J. Thomas, SDNET2018: A concrete crack image dataset for machine learning applications, 2018, http://dx.doi.org/10.15142/T3TD19.
[33] 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, http://dx.doi.org/10.3390/app9142867.
[34] M. Hussain, J.J. Bird, D.R. Faria, A study on CNN transfer learning for image classification, in: Advances in Intelligent Systems and Computing, Vol. 840, Springer, 2019, pp. 191–202, http://dx.doi.org/10.1007/978-3-319-97982-3_16.
[35] S. Niu, Y. Liu, J. Wang, H. Song, A decade survey of transfer learning (2010–2020), IEEE Trans. Artif. Intell. 1 (2) (2020) 151–166, http://dx.doi.org/10.1109/TAI.2021.3054609.
[36] R. Ali, J.H. Chuah, M.S.A. Talip, N. Mokhtar, M.A. Shoaib, Automatic pixel-level crack segmentation in images using fully convolutional neural network based on residual blocks and pixel local weights, Eng. Appl. Artif. Intell. 104 (2021) 104391, http://dx.doi.org/10.1016/j.engappai.2021.104391.
[37] K. Gopalakrishnan, S.K. Khaitan, A. Choudhary, A. Agrawal, Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection, Constr. Build. Mater. 157 (2017) 322–330, http://dx.doi.org/10.1016/j.conbuildmat.2017.09.110.
[38] K. Zhang, H.D. Cheng, B. Zhang, Unified approach to pavement crack and sealed crack detection using preclassification based on transfer learning, J. Comput. Civ. Eng. 32 (2) (2018) 04018001, http://dx.doi.org/10.1061/(asce)cp.1943-5487.0000736.
[39] D. Dais, İ.E. Bal, E. Smyrou, V. Sarhosis, Automatic crack classification and segmentation on masonry surfaces using convolutional neural networks and transfer learning, Autom. Constr. 125 (2021) 103606, http://dx.doi.org/10.1016/j.autcon.2021.103606.
[40] A. Agapiou, K. Adair, D. Meyer, E. Skene, M. Smith, N. Valkov, The development of a smart mobile app for building façade defects inspections, J. Civ. Eng. Archit. 16 (2022) 150–171, http://dx.doi.org/10.17265/1934-7359/2022.03.004.
[41] S. Katsigiannis, S. Seyedzadeh, A. Agapiou, N. Ramzan, Brickwork cracks dataset, 2023, http://dx.doi.org/10.5281/zenodo.8014150.
[42] L. Perez, J. Wang, The effectiveness of data augmentation in image classification using deep learning, 2017, arXiv preprint, arXiv:1712.04621.
[43] C. Shorten, T.M. Khoshgoftaar, A survey on image data augmentation for deep learning, J. Big Data 6 (60) (2019) http://dx.doi.org/10.1186/s40537-019-0197-0.
[44] F. Chollet, et al., Keras, 2015, https://keras.io.
[45] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252, http://dx.doi.org/10.1007/s11263-015-0816-y.

[46] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations, 2015.
[47] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, MobileNetV2: Inverted residuals and linear bottlenecks, in: Proc. IEEE conf. on computer vision and pattern recognition, CVPR, 2018, pp. 4510–4520, http://dx.doi.org/10.1109/CVPR.2018.00474.
[48] C. Szegedy, S. Ioffe, V. Vanhoucke, A.A. Alemi, Inception-v4, inception-ResNet and the impact of residual connections on learning, in: Proc. 31st AAAI conf. on artificial intelligence, 2017, pp. 4278–4284.
[49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proc. IEEE conf. on computer vision and pattern recognition, CVPR, 2016, pp. 2818–2826, http://dx.doi.org/10.1109/CVPR.2016.308.
[50] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proc. IEEE conf. on computer vision and pattern recognition, CVPR, 2017, pp. 1251–1258, http://dx.doi.org/10.1109/CVPR.2017.195.
[51] 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. Mané, 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. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, URL https://www.tensorflow.org/. Software available from tensorflow.org.
[52] D.M. Powers, Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation, J. Mach. Learn. Technol. 2 (1) (2011) 37–63.
[53] M. Narwaria, Does explainable machine learning uncover the black box in vision applications? Image Vis. Comput. 118 (2022) 104353, http://dx.doi.org/10.1016/j.imavis.2021.104353.
[54] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual explanations from deep networks via gradient-based localization, in: 2017 IEEE International Conference on Computer Vision, ICCV, 2017, pp. 618–626, http://dx.doi.org/10.1109/ICCV.2017.74.
[55] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2009, pp. 248–255, http://dx.doi.org/10.1109/CVPR.2009.5206848.
[56] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, Tech. Rep., 2009.
[57] C. Sun, A. Shrivastava, S. Singh, A. Gupta, Revisiting unreasonable effectiveness of data in deep learning era, in: 2017 IEEE International Conference on Computer Vision, ICCV, 2017, pp. 843–852, http://dx.doi.org/10.1109/ICCV.2017.97.
[58] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, 2017, pp. 6000–6010.
[59] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16 × 16 words: Transformers for image recognition at scale, 2020, http://dx.doi.org/10.48550/arXiv.2010.11929, arXiv preprint, arXiv:2010.11929.
[60] G.I. Okolo, S. Katsigiannis, N. Ramzan, IEViT: An enhanced vision transformer architecture for chest X-ray image classification, Comput. Methods Programs Biomed. 226 (2022) 107141, http://dx.doi.org/10.1016/j.cmpb.2022.107141.
[61] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: Hierarchical vision transformer using shifted windows, in: 2021 IEEE/CVF International Conference on Computer Vision, ICCV, 2021, pp. 9992–10002, http://dx.doi.org/10.1109/ICCV48922.2021.00986.
[62] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou, Training data-efficient image transformers & distillation through attention, in: International Conference on Machine Learning, PMLR, 2021, pp. 10347–10357.
[63] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F.E. Tay, J. Feng, S. Yan, Tokens-to-token ViT: Training vision transformers from scratch on ImageNet, in: Proc. IEEE/CVF International Conference on Computer Vision, ICCV, 2021, pp. 558–567, http://dx.doi.org/10.1109/ICCV48922.2021.00060.
[64] C.-F.R. Chen, Q. Fan, R. Panda, Crossvit: Cross-attention multi-scale vision transformer for image classification, in: Proc. IEEE/CVF International Conference on Computer Vision, ICCV, 2021, pp. 357–366, http://dx.doi.org/10.1109/ICCV48922.2021.00041.
[65] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, L. Shao, Pyramid vision transformer: A versatile backbone for dense prediction without convolutions, in: Proc. IEEE/CVF International Conference on Computer Vision, ICCV, 2021, pp. 568–578, http://dx.doi.org/10.1109/ICCV48922.2021.00061.