

- 1 -

A METHOD AND APPARATUS FOR TRAINING A NEURAL NETWORK TO IDENTIFY CRACKS

FIELD OF INVENTION

5

The present invention relates to the field of crack detection on surfaces such as, but not limited to, surfaces of buildings and constructions.

BACKGROUND

10

Inspection of ageing buildings and civil infrastructures is an important task in developed and developing countries. It requires a large amount of human effort. Efficiently monitoring and maintaining such a large number of existing buildings and civil infrastructures is critical yet challenging for both safety and economic reasons. Automatic detection of surface defects (e.g. cracks) in images of building and infrastructure surfaces can potentially relieve the workload of manual onsite inspection, reduce the cost of manpower, and boost maintenance efficiency. In addition, inspection of airplane surfaces and ship surfaces are also important tasks of maintenance in aerospace and shipping industries.

20

SUMMARY

Technical features of the proposed method for identifying cracks on a surface can be summarized as:

25

- A configuration of deep learning models which is composed of a deep convolutional neural network (CNN) for scene recognition (Scene-CNN) and a deep CNN for crack detection (Crack-CNN).
- A method to train deep CNN model (Crack-CNN) with limited sample images of cracks.
- A learning approach which employ uncompleted latent labels to refine the

30

- 2 -

trained Crack-CNN model.

- A Bayesian model integrate the orientation information of local neighbouring detections to filter out false positives.

- 5 Disclosed herein is a method for training a neural network to identify cracks on a surface, the method including the steps of:
- (a) training a neural network to detect a crack; and
 - (b) refining the trained neural network using a direction of the crack.

- 10 The step of (a) training a neural network may comprise:

(a1) receiving at least one training image, each said training image comprising a plurality of pixels; and

(a2) classifying each pixels as either a crack pixel, indicating presence of a potential crack, or a non-crack pixel, indicating no crack present. The step of (a)

- 15 training a neural network may further comprise:

(a3) for each training image of one or more of said at least one training image, selecting a first predetermined number of said crack pixels (selected crack pixels);

- (a4) for each selected crack pixel, generating a patch of crack centred of
20 said selected crack pixel;

(a5) determining whether to retain the patch based on a number of crack pixels in the patch; and

(a6) adding the patch to a training data set if it is determined the patch should be retained. Moreover:

- 25 step or sub-step (a3) may further comprise selecting a second predetermined number of said non-crack pixels (selected non-crack pixels); and

(a4) may further comprise, for each selected non-crack pixel, generating a patch of non-crack centred of said selected non-crack pixel.

- 30 The method may further comprise rotating the at least one image by an angle, and repeating steps (a3) to (a6). The angle may be a random angle.

- 3 -

The step of (a) training a neural network may further comprise:

(a7) training a neural network using the training data set.

5 The neural network may comprise:

at least one convolutional layer;

a latent feature space layer for receiving an output of the at least one convolutional layer;

an output layer for outputting a binary output of crack detected or crack not
10 detected; and

a latent label layer for labelling patches with one or more latent labels based on one or more features identified in the latent feature space layer; and

wherein (a7) training a neural network using the training data set comprises:

applying the at least one convolutional layer and latent feature space layer
15 to the training data set, including extending a convolutional neural network (CNN) to a refined convolutional neural network (R-CNN) for crack patch detection; and

applying the output layer to a first output of the latent feature space layer.
The step of (b) refining the trained neural network using a direction of the detected crack may comprise applying the latent label layer to a second output of the latent
20 feature space layer. The second output may be generated using only patches in which a crack is detected. The second output may instead be generated using only patches determined based on a said selected crack pixel.

The method may further comprise:

25 (c) identifying a crack by applying a Bayesian model for computing the probability of a true crack based on labels obtained from (a) and (b).

Also disclosed herein is a method for identifying a crack, comprising:

(d) receiving at least one further image of a surface; and

30 (e) identifying a crack by:

(e1) applying a neural network (including R-CNN) trained

- 4 -

according to the method described above to the at least one further image;
and

(e2) applying a Bayesian model for computing the probability of a
true crack based on labels obtained from (a) and (b).

5

The Bayesian model may determine crack orientation consistency.

The step of (c) identifying a crack may comprise identifying a string of partially
overlapping patches in which a crack is detected.

10

Also disclosed herein is an apparatus for training a neural network to identify
cracks on a surface, the apparatus including:

memory; and

at least one data processor, the memory comprising instructions that, where

15 executed by the at least one processor, cause the at least one processor to:

perform the methods described above.

Also disclosed herein is an apparatus for detecting cracks on a surface, the
apparatus including:

20 memory; and

at least one data processor, the memory comprising instructions that, where

executed by the at least one processor, cause the at least one processor to:

perform the methods described above.

25 Also disclosed herein is a non-transitory computer readable storage medium
embodying thereon a program of computer readable instructions which, when
executed by at least one data processor of an apparatus for training a neural
network to identify cracks on a surface, cause the apparatus to carry out the
method described above.

30

Also disclosed herein is a non-transitory computer readable storage medium

- 5 -

embodying thereon a program of computer readable instructions which, when executed by at least one data processor of an apparatus for detecting cracks on a surface, cause the apparatus to carry out the method described above.

- 5 In each of the above cases, the surface in which cracks are sought to be detected may be a surface of:
- a building;
 - a construction (e.g. pavement, cement, sidewalk, road or any other constructed surface);
 - 10 an aircraft; and
 - a vessel (e.g. on ocean going vessel, containment vessel, pressure vessel or any other vessel).

DESCRIPTION OF FIGURES

- 15 FIG 1a is a flowchart of a method for training a neural network in accordance with present teachings;
- FIG 1b is a flowchart of a method for identifying a crack in accordance with present teachings;
- 20 FIG 2 is simplified diagram;
- FIG 3 shows the configuration of the proposed crack detection.
- FIG 4 shows an architecture of a crack CNN network.
- FIG 5 shows computation of latent label (orientation code/label) for a positive sample patch.
- 25 FIG 6 shows a refined training of Crack-CNN on both output labels and latent labels.
- FIG 7 shows a table comparing different crack detection methods.
- FIG 8 shows a graph depicting an effect of Bayesian post-processing with different threshold values.
- 30 FIG 9, comprising FIGs. 9(1a) to 9(4d), shows examples of Bayesian post-processing on challenging images.

DETAILED DESCRIPTION

A deep learning approach for crack detection on the surfaces is presented. While the discussion will largely centre around applications of the approach to detecting cracks in surfaces of buildings and constructions, it will be appreciated the same teachings apply to identifying cracks in other types of surfaces such as aircraft and vessels. This approach is designed for applications with an UAV or a robot. This method consists of a scene recognition deep CNN model (Scene-CNN), a crack detection deep CNN model (Crack-CNN), and a Bayesian inference module for post filtering – these modules may be housed in memory 202 or processor 204 of the apparatus 200 of Figure 2. An ImageNet-CNN is retrained to classify the input images as scene for navigation or scene for surface – e.g. building surface – inspection. For the images captured during building surface scanning, the Crack-CNN is applied to scan the images with a detection window of 64×64 pixels. For the local detection patches of potential cracks in each image, the Bayesian model computes the probability of true crack according to the orientations of locally detected crack patches and filters out the false positive detections.

This system is very efficient for real-world applications. Also described herein is refining of the training of deep Crack-CNN by introducing the latent label of crack orientation in addition to the output label of crack (i.e. yes/no). This training approach can significantly enhance the accuracy of the Crack-CNN on real-world images, especially reducing the false alarms in the images.

25

The approaches described herein achieved 95% detection accuracy – much better than conventional approaches like decision trees and SVM.

The proposed deep learning approach for crack detection is presented, which includes system configuration, training of Crack-CNN, refined learning based on latent labels, and Bayesian approach for post filtering. In a first, broad

30

- 7 -

embodiment, the approach involves a method 100 for training a neural network to identify cracks on a surface, as shown in Figure 1a. The method includes the steps of:

- 102: training a neural network to detect a crack; and
- 5 104: refining the trained neural network using a direction of the crack.

Step 102 involves receiving at least one image. The image(s) may be received from an image capture device or other image source 206 as shown in Figure 2. Alternatively, the image(s) may be stored in memory 202 and extracted as
10 required by the processor(s) 204 of apparatus 200.

The images are captured of the surrounds of a device and are used to essentially visually determine trajectory for the device (per the navigation or Scene-CNN) or the presence of cracks on a surface (per Crack-CNN). This vision approach is
15 designed for building and construction inspection by a device such as a drone – e.g. an unmanned aerial vehicle (UAV) – or a robot. As discussed below, in such a task, the images captured by an autonomous mobile agent (a drone or a robot) are used for two purposes: navigation and inspection. To reduce the complexity of training for deep learning models, a two-stage deep learning system is proposed.
20 At the first stage, we employ a Scene-CNN (re-trained ImageNet-CNN) to classify the images for navigation from the images for inspection. The images for navigation capture the comprehensive views of the environment and perspective views of the route for control of movement, and the images for inspection capture the views when scanning the building and construction surface. For images
25 classified for inspection, our Crack-CNN model is applied to scan the image with a detection window. The potential cracks are detected as a string of partially overlapped detection boxes along the cracks in the image. Combining two deep CNN models for different specific tasks, we can reduce the scales of each CNN model. Using small CNN models for separated specific tasks, it becomes easy for
30 training and very efficient for real-time detection. Figure 3 shows the configuration of the proposed crack detection.

To achieve satisfactory performance, a large number of training images with ground truth labels are required for deep learning. However, for the task of building and construction surface inspection, limited sample images are available from open resources, such as Google Images. Moreover, very few, or small, open benchmarking datasets are available from research societies. Directly collecting large images from real world also may not be feasible due to the cost and limitation to access the high-rise buildings, old buildings, and tunnels in infrastructure system.

10

The present methods involve training a deep learning CNN on crack patches. An image is scanned using a small sliding detection window for crack detection. Then, for training, a large number of patches are sampled from each source image. In essence, each patch is taken to comprise a separate image for crack detection. In total, hundreds or thousands of patches can be generated from limited crack images to train deep learning CNN model.

In demonstrating the present method, a dataset of more than 400 colour images of cracks on building and construction surfaces was collected from open sources (e.g. Google Image) and smartphone cameras. The surfaces included cement walls, concrete panels, brick walls, rock walls, painted walls and ground surfaces, and pavements with complex surface texture patterns. To generate ground truth for training and evaluation, given each source image, step 102 further involves step (hereinafter interchangeably referred to as a sub-step) 108 of classifying pixels. The pixel classification generates binary reference image corresponding to the original image, in which white pixels represent cracks and black ones are background surfaces – the tones or colours used to indicate crack and non-crack may be swapped or changed.

Detection windows were then randomly sampled from the training images based on the annotated reference image. For present purposes, to get the balance

- 9 -

between computational cost and accuracy of crack detection, each training sample is 3-channel (RGB) image patch of 64x64 pixels produced based on the following strategies:

- 5 a) Training the neural network based on the binary reference images or ground truths then involved step 110 of selecting crack pixels. The selection process may involve specific, sequential selection steps. However, for present purposes, on each annotated binary reference image 2 percent of crack pixels were randomly selected. Using each of the sampled pixel as a centre, an image
10 patch of 64x64 pixels (i.e. a patch representing a crack) was cropped from the corresponding source image – step 112. All the patches thus generated form the positive training samples.
- b) A similar process was then performed for non-crack pixels – step 108. In particular, when compared with the number of selected crack pixels, four times
15 that number of non-crack pixels were selected from the reference image and patches were then cropped from the corresponding source colour image – per step 112. Some patches are not suitable for use in the method and were therefore culled – step 114. In the present example, a patch in which there are crack pixels within the core circle with diameter of 3 pixels is selected as a positive sample,
20 and a patch of which less than 0.1 percent pixels are crack pixels is selected as a negative sample. If neither case applies, the patch is dropped.
- c) Due to limited crack source images available, the training samples or training data set was augmented using rotated images – step 118. For each selected training patch, the image around the centre point of the patch was rotated
25 by an angle, which may be a random angle of α within $[0^\circ, 180^\circ]$, and a patch again cropped from the rotated image.

All images, or patches, produced using the steps set out above were then added to a training data set – step 116.

30

Out of all generated training patches, 337,867 patches were used as the training

- 10 -

set – step 120 – 84,699 samples were used as the validation set, and an extra 54,873 patches are used as the testing set. The number of crack and non-crack patches were nearly equal in the three data sets.

- 5 A conventional architecture is employed for the neural network, e.g. the ConvNet. The architecture of the Crack-CNN network is depicted in Figure 4. The network consists of at least one, and presently three, convolutional layers, each followed by a max-pooling layer, and 2 fully-connected layers for output. While filters and stride of any suitable dimensions may be used, for present purposes 5×5 filters and 64 feature maps are used for all three convolutional layers, and a 3×3 pooling
- 10 with stride 2 for all the three max-pooling layers. Each layer contains 64 maps. The convolutional layers apply a convolution operation to the input map, all convolutional filter kernel elements being trained from data in a supervised way by learning from the labelled ground truth. After each convolution layer, the max-
- 15 pooling operation is performed to summarize features, so that the network is able to learn spatially invariant features. Then, two fully-connected layers, which connect every neuron in one layer to every neuron in next layer, are used to learn and perform classification. Finally, a soft-max layer is hired to estimate the probability of each class given an input patch.

20

- Assuming there are m training image patches, which are represented as the labelled training set $T = \{x^{(i)}, y^{(i)}\}_{i=1}^m$, where $x^{(i)}$ is the i -th image patch and $y^{(i)}$ is the corresponding class label. The value of $y^{(i)}$ is either '0' or '1', where '1' means a positive sample or a crack patch, and '0' denotes a negative sample or a
- 25 background patch. Let $z_j^{(i)}$ be the output of neuron j in the output layer for patch $x^{(i)}$, then the probability of assigning the label $y^{(i)}$ to patch $x^{(i)}$ can be computed as:

$$p(y^{(i)} = j | z_j^{(i)}) = \frac{e^{z_j^{(i)}}}{\sum_{l=1}^k e^{z_l^{(i)}}}$$

where k is the number of neurons in the output layer. Defined on the standard

cross-entropy loss, the corresponding loss function is:

$$L = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^k y_j^{(i)} \log p(y^{(i)} = j | z_j^{(i)}) = -\frac{1}{m} \left[\sum_{i=1}^m \sum_{j=1}^k 1\{y^{(i)} = j\} \log \frac{e^{z_j^{(i)}}}{\sum_{l=1}^k e^{z_l^{(i)}}} \right]$$

where $1\{\cdot\}$ is the indicator function.

- 5 The Crack-CNN is then refined using latent characters, to extend the CNN to a refined CNN (R-CNN). Deep learning is an approach for end-to-end learning. A learned Deep CNN model can be mathematically expressed as a mapping function $y = F(\mathbf{x}, \theta)$, which maps a high-dimensional input $\mathbf{x}$ to a low-dimensional output label y , where θ represents the large set of parameters in the ConvNet
- 10 model, i.e., the weights connecting the neurons and parameters of the sub-functions of the neurons. The function F represents a many-to-few mapping. With the expected output (label) of an input (image patch), there is uncertainty whether the function represents a correct mapping, as there are uncertainties on what and how the rich features in the high-dimensional raw input are exploited to reach the
- 15 correct label, especially with limited training samples. This is considered as the problem of correctness of predictability for deep learning models, which may be caused by several reasons such as biased training data, overfitting and under fitting of the large scale deep learning models. The incorrect behaviour of predictability may give correct output labels for some training samples, but will
- 20 produce a large number of classification errors when applied to novel inputs. In a manner of sliding window in an image for object detection, as employed for crack detection in the present application, it will produce a large number of false alarms in normal images, resulting in useless for real-world applications. Existing deep learning methods require a large number of manually labelled training samples to
- 25 reduce the uncertainty in the learned many-to-few mapping. However, in many applications, the training sample images may be limited, and expensive human labelling effort might be prohibitive.

With limited training image samples, some more latent information about samples

30 besides the output labels may be helpful to reduce the uncertainty about the

many-to-few mapping. For example, a child may ask for one more question besides the given correct answer when taught image recognition problems, such as “it has four legs for a horse”. Presently proposed is a novel approach for refined deep learning on latent information of some training samples to reduce the uncertainty in the learned many-to-few mapping function, or reduce the possibility of incorrect behaviours in predictability in deep learning models.

The basic idea behind the present approach is to introduce more information of a training sample to constrain the learned many-to-few mapping function more related to the distinctive features of the training sample. For task of crack detection, besides the label of “crack” for each positive sample, we introduce the direction of the crack patch. Hence, the Crack-CNN is trained to predict not only the final label of “crack” but also the direction of the crack, which enhances the correctness of predictability and reduce the uncertainty of the final prediction (i.e. the many-to-few mapping).

For each image patch of crack example, the corresponding ground truth image patch is also cropped, where the white pixels indicate the crack pixels and the black pixels represent the background region, as shown in Figure 5. In each binary ground truth image patch, square least fitting is used to obtain a straight line which best fits the annotated crack (white pixels) in the image patch. From the function of the straight line, the crack direction angle $\beta \in [0, \pi]$ is obtained. The direction angle is evenly quantized into 9 bins from 0 to 8. Each bin covers 20° . Hence, for a special direction value β , its latent label is a binary vector of 9 elements, and one of them is ‘1’ and the rest are ‘0’s. Figure 5 illustrates the computing of a latent label for a positive sample patch. The leftmost image is the image patch of the crack, the next is the binary ground truth image patch, the third image shows the fitted straight line for crack pixels, which is $\pi/4$, and the rightmost image shows the latent label.

Let y_0 denote the output label (i.e., ‘1’ for crack and ‘0’ for non-crack) and y_l

- 13 -

represents the 9-dimensional latent label. Since the latent information is uncompleted (none for negative samples) and inaccurate due to the variation of the crack shapes, a repeat training algorithm is used for refinement of the ConvNet for crack detection. As depicted in Figure 6, and as reflected in the step of refining the neural network 104, the Fc4 layer (fully connected layer 4) is treated as a latent feature space. Firstly, this layer is connected to the output layer so that the output layer receives a first output of the Fc4 layer, to train the Crack-CNN on output labels $\{y_o\}$, as done in the conventional ConvNet. After that, the Fc4 layer is disconnected from the output layer and connected to the layer of latent labels $\{y_l\}$, so that the layer of latent labels receives the output from the Fc4 layer. The Crack-CNN is then retrained on the latent labels for the positive samples of crack patches. This learning step constrains the prediction of a crack as being a crack of desired direction. Next, the Crack-CNN is retrained on the output labels for all the training samples again to reduce potential bias on the positive sample set, or samples produced using a selected crack pixel per step 110. Training is repeated on output labels and latent labels till a high-performance model is obtained – note: in some embodiments, no repetition is necessary.

Figure 6 illustrates a refined training of Crack-CNN on both output labels and latent labels, where solid black arrows indicate feed-forward computing, and dashed black arrows denotes back-propagation learning. The output labels and latent labels are employed repeatedly to train the D-CNN (deep convolutional neural network) model.

Let $S_o = \{x^{(i)}, y_o^{(i)}\}$ be the whole training set with output labels, and $S_l = \{x_p^{(i)}, y_l^{(i)}\}$ be the set of positive samples with latent labels. The training algorithm can be described as the following:

- a) Set $t=1$, randomly initialize a ConvNet as Crack-CNN_{0,2};
- b) Train Crack-CNN_{t-1,2} on S_o and obtain the model Crack-CNN_{t,1};
- c) Train Crack-CNN_{t,1} on S_l and obtain the model Crack-CNN_{t,2};
- d) Set $t=t+1$, repeat b) to d) till $t=T$;

- 14 -

e) Save the best Crack-CNN_{*t*,1}.

where *t* indicates the *t*th iteration of training.

- 5 In training of CNN model, to reduce the effectiveness of overfitting, the dropout strategy is widely used between two fully-connected layers. Normally, the dropout ratio of 0.5 is used. The training process is accelerated by graphics processing units (GPUs). Rectified linear units (ReLU) are also used as the activation function to further speed up the learning process. In testing, the training was implemented
- 10 on server with four Intel Xeon E5-4620 v2 @2.6GHz CPUs and two Nvidia Tesla K40m GPUs. The network was trained using the stochastic gradient descent method with a batch size of 128 patches, momentum of 0.9, and weight decay of 0.0005.
- 15 When performing sliding window approach on an image to detect cracks, many more detection windows cover normal parts of building and construction surfaces when compared with windows containing cracks. Hence, even achieving very high accuracy rate, there would be many false alarms generated on real-world applications. The Crack-CNN produced high-performance results on each
- 20 detection window. The visual classifications are performed independently and spatial isolated. Here, a Bayesian inference is proposed for application on local neighbourhoods to filter out false alarms.

Let **x** denote the location of a crack detected by Crack-CNN, *N_x* represents a local
 25 neighbourhood of detection at **x**, and **x_j** be one of other detections within *N_x*. If the detection at **x** is correct, there should be a string of detections along the crack; otherwise, the neighbouring detections would be few and appear randomly within the neighbourhood with a wide variation of estimated directions. Since the Crack-CNN is trained on both output class label and the latent label of direction code,
 30 both the prediction of crack (probability (1)) and the prediction of direction code (latent label *y_l*) can be obtained. Exploiting the consistency of orientation of local

- 15 -

detections, the probability of detection at $\mathbf{x}$ being a true crack based on its association with neighbouring detections can be expressed as:

$$P(y = 1|y_o, y_l, N_x) = \frac{1}{K} \left(P(y_o = 1|z_1) + \sum_{\mathbf{x}_j \in N_x} P(y_o^j = 1|z_1^j) P(\mathbf{x}_j|\mathbf{x}) P(y_l^j|y_l) \right)$$

5

where $P(y_o = 1|z_1)$ denotes the probability of classifying the patch as crack, $P(\mathbf{x}_j|\mathbf{x})$ represents the spatial proximity of $\mathbf{x}_j$ to $\mathbf{x}$, and $P(y_l^j|y_l)$ represents the consistency of directions between crack segments $\mathbf{x}_j$ to $\mathbf{x}$. $P(y_o = 1|z_1)$ comes from the output of Crack-CNN (1). The spatial proximity can be computed as:

10

$$P(\mathbf{x}_j|\mathbf{x}) = e^{-\|\mathbf{x}_j - \mathbf{x}\|^2 / (2\sigma_s^2)}$$

Suppose the n -th bin of y_l is '1', and the m -th bin of y_l^j is '1'. Then, the orientation consistency can be computed as:

15

$$P(y_l^j|y_l) = e^{-\min(|n-m|, 9-|n-m|)}$$

With fixed size of the neighbourhood, the decision can be made with a high confidence that the detection is truly a patch containing a crack if

20 $P(y = 1|y_o, y_l, N_x)$ is larger than a pre-set threshold.

Once the neural network is trained and refined, it can be used in practice, as shown in Figure 1b. Per step 122, the method 100 may involve receiving at least one further image of a surface. Presently a number of further images are received and classified per step 124 as described above, into navigation images to which Scene-CNN is applied, and crack detection images to which Crack-CNN is applied. For the navigation images, they are used to determine a trajectory for the device (e.g. drone or robot) in a known way. For the crack detection images,

25

- 16 -

processes are performed to identify cracks – step 126. In particular, the refined Crack-CNN is used – step 128 – to identify crack patches – step 130 – and their orientation labels, and the Bayesian model is then applied – step 132 – to determine which cracks are true cracks based on their being part of a string of
5 partially overlapping patches in which a crack is detected, which may include determining if the positions of the crack patches are sufficiently aligned so that the cracks can be considered parts of the a single crack continuing for multiple patches – step 134.

10 Figure 2 illustrates an apparatus 200 for performing the method 100. The apparatus 200 includes memory 202 and at least one processor 204. The memory 202 contains instructions, and may also contained the images from which the training data or image set is produced. The instructions are executed by the processor(s) 204 and cause the processor(s) 204 performed steps 102 to 120 –
15 certain steps may be omitted in some embodiments – to train the neural network and also to perform step 122, and/or to perform steps 124 to 130 in applying a trained neural network to a new set of images.

The memory may be a non-transitory computer readable storage medium
20 embodying the method 100, or various steps of the method 100, thereon.

After training the neural network, in the evaluation on test images, the labelling strategy is slight different. For a detected window, the patch is labelled as *true positive* (TP) if that contains crack pixels within its central 32x32 region, otherwise,
25 it is labelled as *false positive* (FP). For a detection window of negative output, we label it as *false negative* (FN) if it contains crack pixels within its central 32x32 region, otherwise, it is labelled as *true negative* (TN). To achieve a comprehensive estimation of a model's performance for crack detection on test images, three well-used statistic metrics are applied: precision, recall and F1 score:

- 17 -

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F1 = \frac{2PR}{P + R}.$$

In one test experiment, the present method was compared against the method adopted in convolutional neural network (ConvNet) method as both of the dataset and the model are available online. A 5-fold cross-validation is performed. For a fair comparison, we do not use Bayesian post-processing in this experiment.

Since the ground truth pixels are not available in the dataset from conventional methods, they cannot be used for refined training on latent labels. The samples from this dataset are related simple compared to our dataset since they are all collected from pavement images of cement ground surfaces. First, the training set is employed to train Crack-CNN on y_o , and then the present dataset is used to train Crack-CNN on y_l . The two training processes are repeated as aforementioned. Finally, the trained Crack-CNN model is evaluated on the test set from conventional methods. The results are listed in Table 1 as model "Proposed model (1)". It can be seen that the refined training on latent labels is able to clearly improve the final classification performance.

On review of the samples of the dataset from conventional work, it was found that there are many uncertain and mislabelled samples. Then, the labels for the dataset were refined. After that, the dataset with refined labels was used to train Crack-CNN on y_o , and the present dataset was used to train Crack-CNN on y_l as done in the first test. In this time, much better performance was exhibited as listed in Table 1.

Table 1. The experiment results on the dataset

Method	Precision	Recall	F1 score
SVM	0.8112	0.6734	0.7359
Boosting	0.7360	0.7587	0.7472
ConvNet	0.8696	0.9251	0.8965

Proposed model (1)	0.8962	0.9295	0.9125
Proposed model	0.9627	0.9803	0.9714

Next, the present model was compared with ConvNet using the presently developed dataset, which contains substantially more challenging samples from various building and construction surfaces. After building a training set, another image set was collected for testing – the same applies to the 'use' case, involving receiving one or more further images for classification at step 124 –, which contains some images quite different with those in training set. This testing is more challenging than the normal testing of which both the training set and the testing set are randomly selected from a same dataset. ConvNet was trained on the present training set just using the class label y_o , and the Crack-CNN model was trained on both class label y_o and latent label y_l . The performance of ConvNet and Crack-CNN on the test set is listed in Table 1. As the present dataset covers a much wider range of diversity, the classification task became very hard. Hence, all the first three metric values are lower than ConvNet dataset. As we can see from Table 1, F1 score of the present method is much larger than ConvNet's model. It is worth noting that, since the refined training on latent labels is only performed on positive samples, it does not affect recall rate, however, it improves the precision significantly (increased over 13%). This observation supports the assumption that employing the latent labels in training may enhance the correctness of predictability and reduce the uncertainty of the final prediction (i.e. the many-to-few mapping) for deep learning models. When applying sliding window on an image for crack detection, most of the detection windows are non-crack patches. The proposed Crack-CNN model can reduce about half of the false positive detections (49.4%).

25

Table 2. The experiments on the present dataset.

Method	Precision	Recall	F1 score	Accuracy
ConvNet	0.6631	0.5924	0.6358	0.9160

- 19 -

Proposed model	0.7936	0.5858	0.6741	0.9326
-------------------	--------	--------	--------	--------

To visually evaluate the performance of crack detection on images with sliding window approach, the methods used in ConvNet were compared to the results from dense scanning detections. The sliding window size is 64x64 pixels, and the
 5 step size is 16 pixels. Some results are shown in Figure 7, where the boxes indicate the detected crack patches. It can be seen that the present method generates much fewer false alarms.

The effect of Bayesian Inference for post-processing is presented.

10

For each patch detected by CNN, support was evaluated from a 12x12 neighbourhood. For each pair of the detected patch and a neighbouring patch, the present method evaluated:

- The distance as the weight to accumulate the evidence;
- 15 • The consistency of crack directions;
- The consistency of location and crack direction.

Accumulation over the 12x12 neighbourhood, removed the detected patches with less local support. On test images with many false positives from CNN model, the
 20 Bayesian post-processing can remove more than 60% of false positives with no significant loss on positive detections, or more than half of the false positives were removed.

The effect of Bayesian post-processing with different threshold values is shown in
 25 Figure 8. The threshold value of 0.7 results in the best performance.

Further examples of Bayesian post-processing on challenging images are shown in Figures 9(1a) to 9(4d), where all the patches are the detections from CNN, the grey boxes are the false positives and removed by Bayesian inference, and the

- 20 -

solid black boxes are final detections confirmed by the Bayesian inference. In particular, Figures 9(1a), 9(2a), 9(3a) and 9(4a) each show an unprocessed image of a crack. The present method (i.e. as shown in Figure 1b) is applied. Crack patches are identified as shown in Figures 9(1b), 9(2b), 9(3b) and 9(4b). The crack patches follow the crack and are identified at least partially based on a string of neighbouring patches with common bin, or neighbouring bin (e.g. $y = [0,1,0,0,0,0,0,0]$ being a neighbour to $y = [1,0,0,0,0,0,0,0]$ and $y = [0,0,1,0,0,0,0,0]$), designating similar crack direction as discussed above. Non-crack patches are also identified as shown in Figures 9(1c), 9(2c), 9(3c) and 9(4c). The total set of crack and non-crack patches is shown in Figures 9(1d), 9(2d), 9(3d) and 9(4d).

It will be appreciated that many further modifications and permutations of various aspects of the described embodiments are possible. Accordingly, the described aspects are intended to embrace all such alterations, modifications, and variations that fall within the spirit and scope of the appended statements.

Throughout this specification and the statements which follow, unless the context requires otherwise, the word “comprise”, and variations such as “comprises” and “comprising”, will be understood to imply the inclusion of a stated integer or step or group of integers or steps but not the exclusion of any other integer or step or group of integers or steps.

The reference in this specification to any prior publication (or information derived from it), or to any matter which is known, is not, and should not be taken as an acknowledgment or admission or any form of suggestion that that prior publication (or information derived from it) or known matter forms part of the common general knowledge in the field of endeavour to which this specification relates.

CLAIMS

1. A method for training a neural network to identify cracks on a surface, the method including the steps of:
- 5 (a) training a neural network to detect a crack; and
(b) refining the trained neural network using a direction of the crack,
wherein (a) training a neural network comprises:
- (a1) receiving at least one training image, each said training image comprising a plurality of pixels; and
- 10 (a2) classifying each pixels as either a crack pixel, indicating presence of a potential crack, or a non-crack pixel, indicating no crack present,
wherein, for each training image of one or more of said at least one training image,
(a) training a neural network comprises:
- (a3) selecting a first predetermined number of said crack pixels (selected
- 15 crack pixels);
- (a4) for each selected crack pixel, generating a patch of crack centred of said selected crack pixel;
- (a5) determining whether to retain the patch based on a number of crack pixels in the patch;
- 20 (a6) adding the patch to a training data set if it is determined the patch should be retained; and
- (a7) training a neural network using the training data set,
wherein the neural network comprises:
- at least one convolutional layer;
- 25 a latent feature space layer for receiving an output of the at least one convolutional layer;
- an output layer for outputting a binary output of crack detected or crack not detected; and
- a latent label layer for labelling patches with one or more latent labels based
- 30 on one or more features identified in the latent feature space layer; and
wherein (a7) training a neural network using the training data set comprises:

applying the at least one convolutional layer and latent feature space layer to the training data set, including extending a convolutional neural network (CNN) to a refined convolutional neural network (R-CNN) for crack patch detection; and applying the output layer to a first output of the latent feature space layer.

5

2. A method according to claim 1, wherein:

(a3) further comprises selecting a second predetermined number of said non-crack pixels (selected non-crack pixels); and

(a4) further comprises, for each selected non-crack pixel, generating a patch of non-crack centred of said selected non-crack pixel.

10

3. A method according to claim 1 or 2, further comprising rotating the at least one image by an angle, and repeating steps (a3) to (a6).

15 4. A method according to claim 3, wherein the angle is a random angle.

5. A method according to any one of claims 1 to 4, wherein (b) refining the trained neural network using a direction of the detected crack comprises applying the latent label layer to a second output of the latent feature space layer.

20

6. A method according to claim 5, wherein the second output is generated using only patches in which a crack is detected.

25 7. A method according to claim 5, wherein the second output is generated using only patches determined based on a said selected crack pixel.

8. A method according to any one of claims 1 to 7, further comprising:

(c) identifying a crack by applying a Bayesian model for computing the probability of a true crack based on labels obtained from (a) and (b).

30

9. A method for identifying a crack, comprising:
- (d) receiving at least one further image of a surface; and
 - (e) identifying a crack by:
 - (e1) applying a neural network trained according to the method of
 - 5 any one of claims 1 to 8 to the at least one further image; and
 - (e2) applying a Bayesian model for computing the probability of a true crack based on labels obtained from (a) and (b).
10. A method according to claim 8 or 9, wherein the Bayesian model determines
- 10 crack orientation consistency.
11. A method according to any one of claims 8 to 10, wherein (c) identifying a crack comprises identifying a string of partially overlapping patches in which a crack is detected.
- 15
12. A method according to any one of claims 1 to 11, wherein each said surface is a surface of at least one of:
- a building;
 - a construction;
 - 20 an aircraft; and
 - a vessel.
13. An apparatus for training a neural network to identify cracks on a surface, the apparatus including:
- 25 memory; and
- at least one data processor, the memory comprising instructions that, where executed by the at least one processor, cause the at least one processor to:
- perform the method of any one of claims 1 to 7, or claim 12 when dependent on any one of claims 1 to 7.

14. An apparatus for detecting cracks on a surface, the apparatus including:
memory; and

at least one data processor, the memory comprising instructions that, where
executed by the at least one processor, cause the at least one processor to:
perform the method of any one of claims 8 to 12.

5

15. A non-transitory computer readable storage medium embodying thereon a
program of computer readable instructions which, when executed by at least one
data processor of an apparatus for training a neural network to identify cracks on a
surface, cause the apparatus to carry out a method according to any one of claims

10 1 to 7, or claim 12 when dependent on any one of claims 1 to 7.

16. A non-transitory computer readable storage medium embodying thereon a
program of computer readable instructions which, when executed by at least one
data processor of an apparatus for detecting cracks on a surface, cause the
15 apparatus to carry out a method according to any one of claims 8 to 12.