Multi-level feature fusion in densely connected deep-learning architecture and depth-first search for crack segmentation on images collected with smartphones

Article *in* Structural Health Monitoring · January 2020

DOI: 10.1177/1475921719896813

Multi-level Feature Fusion in Densely Connected Deep Learning Architecture and Depth First Search for Crack Segmentation on Images Collected with Smartphones

[1]Qipei Mei, [2]Mustafa Gül
[1] PhD Student
[2]Associate Professor (Corresponding Author: mustafa.gul@ualberta.ca)
Department of Civil and Environmental Engineering, University of Alberta
Edmonton, Alberta, T6G 2W2, Canada

Abstract

Cracks are important signs of degradation in existing infrastructure systems. Automatic crack detection and segmentation plays a key role in developing smart infrastructure systems. However, this field has been challenging over last decades due to irregular shape of the cracks and complex illumination conditions. This paper proposes a novel deep learning architecture for crack segmentation at pixel level. In this architecture, one convolutional layer is densely connected to multiple other layers in a feed-forward fashion. Max pooling layers are used to reduce the dimensions of the features, and transposed convolution layers are used for multi-level feature fusion. A depth first search based algorithm is applied as post-processing tool to remove isolated pixels and improve the accuracy. The method is tested on two previously published datasets. It can reach 92.02%, 91.13% and 91.58% for precision, recall and F1 score for the first dataset, and 92.17%, 91.61% and 91.89% for the second dataset. The performance of the proposed method outperforms other state-of-the-art methods. At the end of the paper, the influence of feature fusion methods and transfer learning are also discussed.

Keywords: crack detection; deep learning; transposed convolution layer; densely connected layers; depth first search

1. Introduction

The demand for assessing the condition of existing infrastructure is increasing as they approach their designed lives.[1, 2] Cracks on surface of building materials, such as concrete or asphalt, are early signs for the potential damage in the infrastructure which serves as a good indicator to assess the current condition of infrastructure. Propagating defects in the life circle of the infrastructure may cause serious safety issues if they are not treated in timely manner. Current practice in defects detection is primarily visual inspection-based, with high costs and low efficiency, and thus such defects as cracks may be present for a considerable amount of time before they are repaired.

Due to the abovementioned reasons, the demand for automating the crack detection procedure is increasing in these years.[3-5] Efforts have been made in this research field. The commonly used data include vibration signals, ultrasound data, camera images, etc.[6-8] Image based crack detection techniques are among the most promising ones for full automation of this procedure.[9] In fact, image based methods have been utilized in various aspects of infrastructure monitoring and condition assessment.[10, 11] Some of the challenges about image-based crack detection methods include irregular shape and varying size of the crack and the uncertainty in illumination conditions. In past years, the development of deep learning technologies has

attracted attention from researchers in civil engineering. Deep learning has been successfully applied to various tasks including image classification and segmentation.[12-15]

The feasibility of applying deep learning onto image or patch level automatic crack detection on various infrastructure systems, such as buildings[13], road pavements[16, 17], nuclear plants[15, 18] and tunnels[19], has been verified by different researchers recently. The automatic crack detection results can be integrated into infrastructure management system for further decision making. Even though some methods have achieved good performance at image or patch levels, the pixel level accuracy is still an open challenge in this field. Further information such as width, length and orientation of cracks can be accurately extracted only when the cracks can be identified accurately at pixel level. Also, due to the limited data regarding cracks in the field, the generalization of the methods, which means training the method on one dataset and applying it to another, is also a challenge.

In this paper, a novel systematic method including a densely connected deep neural network and a depth first search (DFS) based algorithm for post-processing is introduced for pixel-level crack segmentation. The method is termed as DenseCrack in this paper for simplicity. Unlike other deep neural networks for crack detection methods, where layers are sequentially connected[13, 20, 21], DenseCrack connects each convolutional layer with every other layer in a feed-forward fashion so that later layers could make better use of low-level features. In this method, a DFS based algorithm is also applied as a post-processing step to improve the performance of the deep neural network.

The contribution of this paper includes the following: 1) Skip connections are included in the deep neural network to directly transmit early layer features to later layers to maintain the high frequency content of the image and make the boundaries of cracks clear; 2) We propose a DFS based algorithm to post-process the deep neural network output to overcome the limitations of transposed convolutional layers and to prevent the stains being mistakenly identified as cracks; 3) The method achieves highest performance on publicly available datasets among all the state of the art algorithms; 4) The working principle of the proposed method on crack detection is investigated, and its generalization is evaluated when the proposed method is trained on one dataset and apply to another one.

This paper is organized as follows. Section 2 reviews related literature for crack detection. Section 3 explains the proposed deep neural network and the DFS based algorithm in detail. Section 4 shows the results and discuss the influence of transfer learning and feature fusion methods. Also, the generalization of the method on another dataset is discussed at the end of section 4. Conclusions are drawn in section 5.

2. Related Work

2.1. Traditional Image Processing Methods

At the early age of this research field, researchers investigated the feasibility of filter-based methods on crack detection. Abdel-Qader et al.[22] made a comparison among four edge detection techniques i.e., fast Haar transform, fast Fourier transform, Sobel and Canny operators. In their paper, they identified the fast Haar transform is more reliable in crack detection than other techniques. Sinha et al.[23] proposed a three-step approach in which two statistical filters are first applied to the images, features extracted are merged and then the detection results are cleaned and linked to provide final results.

The major challenge of the filter-based methods is that crack is a global property of an image, the filter-based methods tend to determine the crack based on local features which could be significantly affected by

the noise such as varying lighting conditions. To overcome these issues, researchers have applied different techniques for eliminating the effect of varying conditions.[24-27] In the method proposed by Fujita and Hamamoto[28], they used a median filter to remove shadings from concrete surface images and a line filter with Hessian matrix to enhance cracks, and then used probabilistic relaxation for crack detection. Iyer and Sinha[25] found that cracks in pipe images are darkest in the images, locally linear and have tree-like geometry. Based on their observation, they designed an algorithm consisting of cross-curvature evaluation and linear filtering. The above two methods mainly considered the difference of cracks and background in terms of pixel intensity. The methods could not distinguish the cracks from shadows because both of them had very similar levels of pixel intensity. Adhikari et al.[29] developed a model consisting of crack quantification, change detection, neural networks, and 3D visualization models to digitalize the cracks on bridges. The work shown in this paper was conducted in a controlled environment with clean concrete surface and no illumination changes, which is different than the real life applications. Turkan et al.[30] proposed an approach based on adaptive wavelet neural network, and applied it to terrestrial laser scanners data for crack detection. Compared to camera images, depth information was captured by the laser scanners. Laser scanner is more tolerant to illumination changes but is still suffering from the surface roughness.

Despite the efforts, the variation of illumination conditions and background still makes this problem very difficult for traditional image processing methods. Most of the studies were conducted on certain datasets. Features were handcrafted and finetuned to reflect the properties of the given datasets. However, the crack detection problems are more difficult in real life situations, and the influence of illumination condition, material and even imperfection in imaging system could play important roles in the success of the methods. It is almost impossible to manually design some simple features, which can work for various complex situations. Therefore, in recent years, researchers tend to seek to a more automatic and robust way for crack detection, i.e., deep learning.

2.2. Deep Learning Methods

Comparing with traditional image processing methods, deep learning-based methods offer significant advantages. First, they can take into consideration of the varying lighting conditions and irregularity of crack shape as long as training data includes such varying conditions. Second, the whole feature extraction process is automatic, and no effort need to be made to handcraft features for crack detection. Third, the performance of deep learning-based methods is typically better than traditional methods in terms of crack detection.

Protopapadakis et al.[31] introduced an integrated automatic platform for road and railway tunnel inspection. Based on this platform, a 3-layer convolutional neural network taking 9×9 patches as input was used as feature extract. A multi-layer perceptron was applied for crack detection task using these features. In 2017, Cha et al.[13] introduced convolutional neural network (CNN) for crack detection. They trained a deep CNN on 40,000 images with a resolution of 256×256 pixels and validated 55 images with a resolution of 5,888×3584 pixels. They showed CNN is good at detecting thin cracks under varying lighting conditions and is more robust to noise than any traditional methods. As examples of the early deep learning based methods, Protopapadakis et al.[31] and Cha et al.[13] generated a single label for a patch, and sliding window techniques were applied to scan through the whole image. This kind of techniques could only provide information about whether a patch include cracks or not. The patch is generally much larger than the width of cracks, and the

details of the cracks would be lost in these methods. There are also other studies regarding patch-wise crack detection.[16, 32-35]

To overcome the abovementioned limitations, researchers tend to seek pixel-level crack detection methods to extract more detailed information about cracks. Zhang et al.[36] developed a deep neural work called CrackNet which takes the feature maps extracted by line filters as input and outputs the pixel level prediction of the crack. They showed that the proposed method can outperform traditional methods in terms of F1 score. Ni et al.[37] proposed a method consisting of two neural networks. The first neural network is used for feature extraction, and the second one is for pixel level crack classification. In 2019, Dung and Anh[21] developed a fully convolutional neural network for crack detection using VGG16 as encoder, and tested their method on images extracted from a video about a cyclic loading test on a concrete specimen. Bang et al.[20] introduced an encoder-decoder neural network integrating residual blocks for pixel-level crack detection and applied it on road images collected from a dash camera on moving vehicles. Other studies under this field can be found here. Other pixel-level crack detection methods can be found here.[2, 38]

The current pixel-level crack detection methods usually applied a chain-like architecture in an encoder-decoder fashion. The features were downsampled first and then upsampled. One defect of such setting was that the high frequency (detailed) information could be lost during the downsampling, and cannot be recovered from upsampling. Also, the transposed convolution layers did not take advantage of the information from adjacent pixels, which were likely to lead to sparsely distributed output. The inaccurate sparsely distributed output would make it difficult to further extract the properties of the cracks, such as width, length and orientation, which are important to assess the condition of structures.

In this paper, we focus on addressing the abovementioned issues for pixel level crack detection. We first apply massive skip connections among layers to make additional paths to transmit early layer features to later layers directly. In this way, the detailed information of the input can be kept. To resolve the issue about sparsely distributed output, a DFS algorithm is introduced to find all the connected components in the output, and filter out the ones that have a small number of pixels.

3. Methodology

The proposed method includes two parts, i.e. a densely connected deep neural network and a depth first search based post-processing algorithm. For the efficiency of training and testing, the input image will be split into 128×128 patches and then be fed into the neural network. The prediction of the neural network is a binary mask with the same size as input patch. Then, the patches are integrated together to form a binary mask with the size of original image. Afterwards, the DFS based algorithm is applied to the binary mask to generate a final prediction of the cracks. The neural network will be explained in section 3.1, and the DFS based algorithm will be in section 3.2.

3.1. The densely connected deep neural network

3.1.1. Overall Architecture

In this paper, a neural network with densely connected convolutional layers and transposed convolution layers is proposed. In this architecture, the densely connected convolutional neural network serves as feature extractor, and transposed convolution layers are used for multi-level feature fusion. The advantages of the

proposed neural network compared to other deep neural networks include: 1) parameters are reused in multiple layers so that better performance can be achieved with fewer parameters; 2) with the help of dense connections, a deeper neural network can be trained efficiently; 3) the transposed convolution layers can reserve the spatial features of the image.

In Figure 1~~Figure 1~~, the overall architectures of three variations of the proposed neural network are presented. The only difference among them is the number of layers, where DenseCrack121 is the shallowest with 121 convolutional layers as feature extractor and DenseCrack201 is the deepest with 201 convolutional layers. The difference of number of layers among these variations lies in groups Dense3 and Dense 4 in Figure 1~~Figure 1~~.

Taking DenseCrack121 as an example, the input of the network is 128×128×3 color image patch. The patch is first passed through a 7×7 convolutional layer with depth of 64 and stride of 2. Then, max pooling layer is applied to the output of the first layer to generate a feature map of 32×32×64. Afterwards, a series of dense blocks are applied. Each dense block includes a number of basic components consisting of a 1×1 and 3×3 convolutional layers. Each basic component will be connected to any components following it in the dense block. More details of this block will be explained in section 3.2.3. The dense block does not change the length and width of feature maps (depth is changed), because this function is implemented by a transition block with a convolutional layer and average pooling layer. The output of the last densely connected layer has a size of 4×4. Transposed convolution layers (see section 3.2.4) are applied on it to upsample to an 8×8 feature map. This feature map is fused with an intermediate output from Dense3. After that, the fused feature map is upsampled again to 16×16. Similarly, the 16×16 feature map is fused with output from Dense2 to get information from an early layer. Finally, the feature map integrating information from different layers is upsampled 8 times to 128×128 which is the size of the original patch to generate a binary mask for cracks.

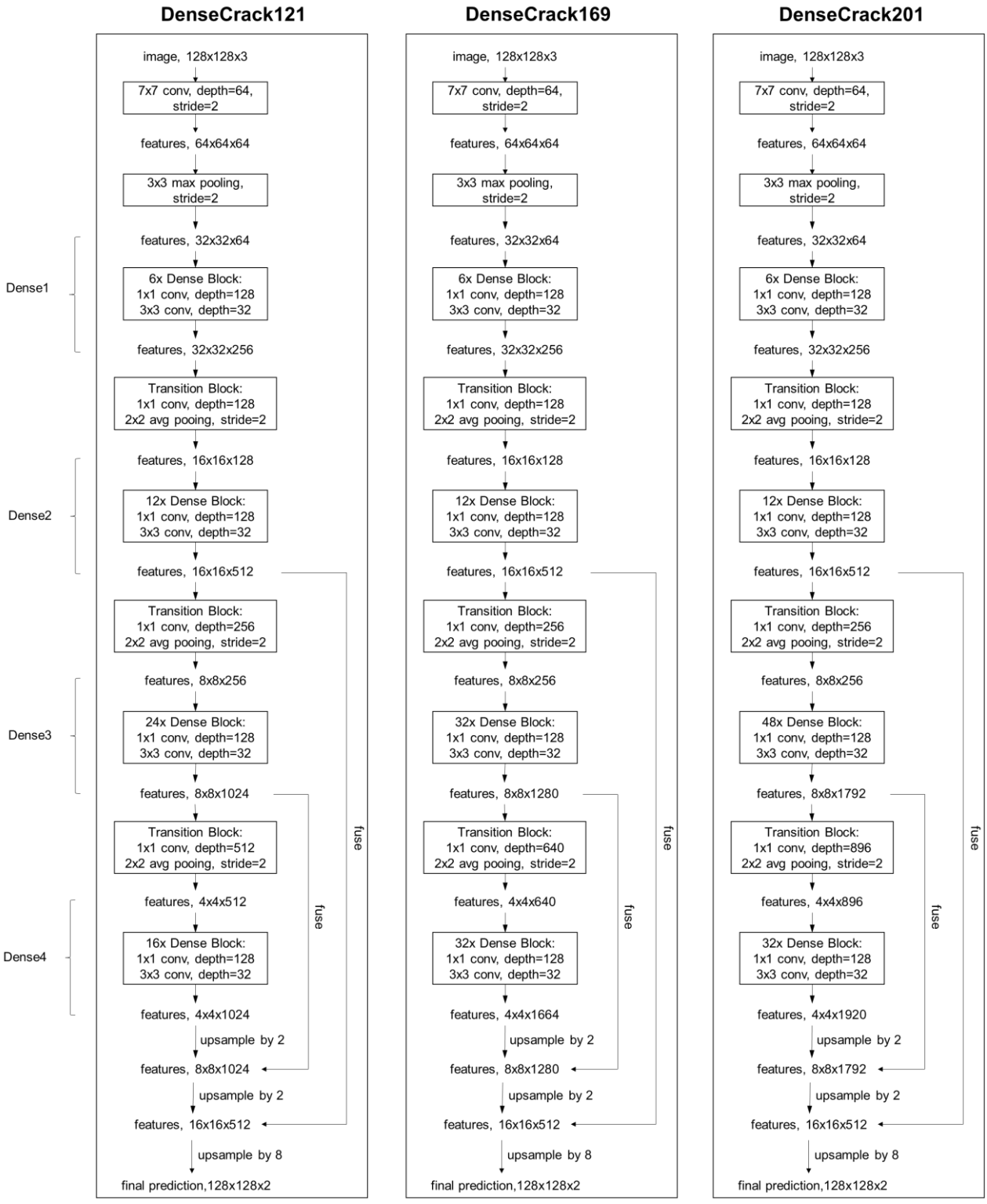


Figure 1 – Architecture of the deep neural network

3.1.2. Convolutional layer

Convolutional layers were first proposed by LeCun et al.[39] The operation of a convolutional layer is very similar to a regular filter in traditional image processing techniques except that the weights of a convolutional layer are determined by the training process in the deep neural network. Due to this characteristic, convolutional layers are suitable for image processing, and has been tremendously successful in computer vision applications.[12, 40] Fundamentally, a convolution layer is the combination of a series of linear operations. The main difference between a convolutional layer and a standard hidden layer in traditional neural network is that it replaces the multiplication between weights and inputs with a convolution operation.[41] The convolution operation can be represented as below,

$$I_{out}(i,j,k) = \sum_m \sum_n I_{in}(i-m, j-n, l) w_k(m,n,l) + b_k$$

where I_{in} and I_{out} stand for the input and output features of the convolutional layer, and w_k and b_k are the weights and bias for the layer.

In the practical implementation, a convolutional layer can be expressed as the multiplication of matrices with certain patterns. Figure 2~~Figure 2~~ shows an example of converting a convolutional operation to matrix multiplication. For a 2×2 filter and 3×3 image, the convolution matrix is 4×9 and the image can be expanded to 9×1. The resulting matrix is 4×1, which can be then reshaped to 2×2. It is clear that convolutional is still matrix multiplication. The main difference is that the weight matrix is sparse with shared values and follows a certain pattern. Therefore, the convolutional layer needs less space for storage and has less parameters to estimate.

In the proposed neural network, downsampling is achieved by convolution layers with stride of 2 and a max pooling layer. As can be seen in Figure 1~~Figure 1~~, the first convolutional layer is followed by a max pooling layer. This layer scans over the sub-regions of the feature space and extract the maximum values of them. With a stride of 2, the max pooling layer can downsample the features by a factor of 2.

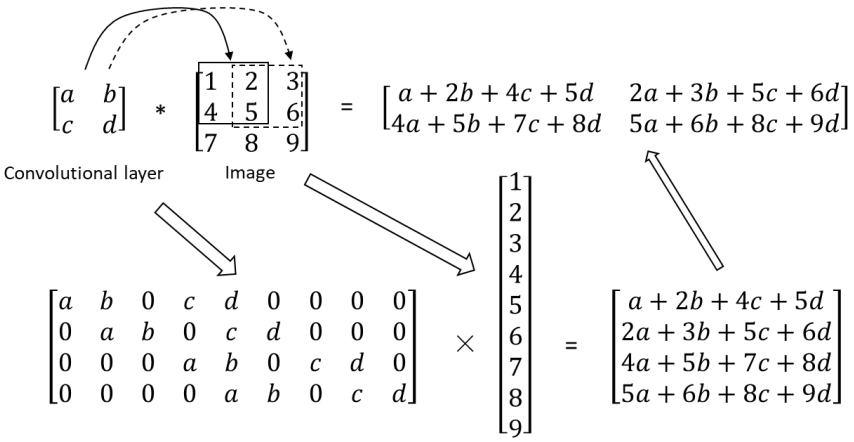


Figure 2 – Matrix multiplication form of a convolutional layer

3.1.3. Dense Block

In previous studies, it is believed that deeper CNNs would lead to a boost in performance.[42] This could be explained by Figure 3~~Figure 3~~. Considering a shallow CNN and its deep counterpart, there are two scenarios: 1) if the mapping function we are looking for is simple enough, the deep CNN is equivalent to its shallow counterpart and an identity function (H in the figure); 2) if the mapping function is complex, the

deeper CNN should have better approximation to it. However, in practice, simply adding more blocks to the existing CNNs would cause problems for training which further degrade the performance.[43] The reason is that more parameters from non-linear layers have to be estimated to learn the same mapping in deeper CNN, which may cause gradient vanishing and under fitting.[44]

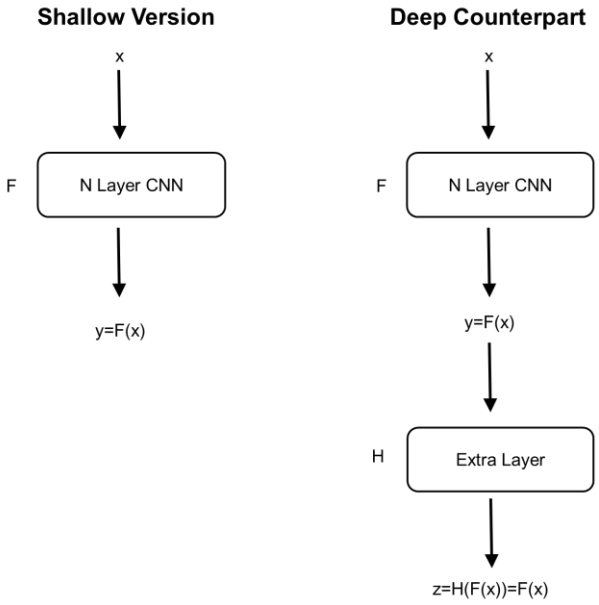


Figure 3 – Comparison of shallow and deep CNN

To resolve the training issue related to deeper CNN and gain the benefits of performance boost, He et al. introduced skip connections between non-adjacent layers to the architecture of the CNN.[43] In their neural network, the block with skip connections is called residual block. Huang et al.[45] moved one step further and proposed a dense block in which every layer is connect to every other layers in a feed-forward fashion. With the help of dense block, the error signal from back-propagation can be easily transmitted to early layers. This can help the training of the neural network to gain better performance. The neural network with dense blocks is usually termed as DenseNet.

A typical 6-time dense block, i.e. Dense1 in Figure 1~~Figure 1~~, is presented in Figure 4~~Figure 4~~. It is seen that the output from the previous layer is fed into the block as an input. There is a forward stream composing of convolutional layers. Also, each component is connected to all the components following it within the block. Taking component 1 as an example, the input (32×32×64) is passed through a 1×1 convolution with depth of 128 and 3×3 convolution with depth of 32 to generate component 1 (32×32×32). Then, the input is concatenated with component 1 through skip connection to generate a concatenated feature map of 32×32×(64+32). Similarly, the concatenated feature map (32×32×96) is used to generate component 2 (32×32×32) using convolutional layers. This time, features from both input and component 1 are concatenated with component 2 features to generate a feature map of 32×32×(64+32+32). It is clear that the feature map has an increase of 32 in depth when one more component is applied. Repeating the procedure for all the components, the final feature map generated from Dense1 block is 32×32×(64+32×6)=32×32×256. Similar calculations are used to determine dimensions for all other dense blocks in the proposed neural network.

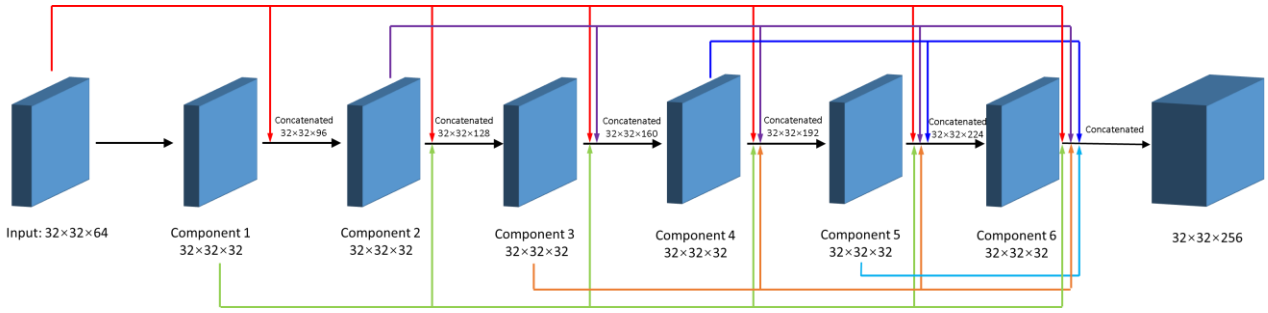


Figure 4 – Details of Dense1 block

3.1.4. Transition Block

It is seen in last sections that the dense block only changes the depth of the features but not the length and width. The reduction of length and width of features are implemented by a transition block. This block consists of a 1×1 convolutional layer with half the depth of the input and a 2×2 average pooling layer with a stride of 2. The convolutional layer reduces the depth to half for computational efficiency and average pooling layer reduces the length and width of features to half by picking average of 4 adjacent elements.

3.1.5. Transposed convolution layer

As mentioned in the previous section, the transition block reduce the height and width of features by half. Therefore, the size of features becomes smaller and smaller as the neural network goes deeper. Since the final feature map that we need is the same size as the original image, upsampling is necessary. Regularly, interpolation techniques are applied for upsampling. However, this is manual feature engineering, during which information may be lost. In this paper, transposed convolution layer is implemented to conduct upsampling. This layer was first time used for upsampling in Long et al.[46] and gained excellent performance in image segmentation.

Similar to a convolutional layer, the transposed convolution layer is explained using an example as presented in Figure 5~~Figure 5~~. In transposed convolution operation, every single pixel is multiplied by the whole 2×2 filter which forms a 2×2 block to be placed in corresponding locations in the output matrix. The upsampling is implemented by placing all the blocks into right locations. Like convolution operation, transposed convolution operation can also be written as matrix multiplication form. In this example, it is written as 9×4 matrix multiplied by 4×1 matrix. The resulting 9×1 matrix can then be reshaped to 3×3 matrix. It should be noted that transposed convolution is not inverse operation of convolution, so it cannot be used to recover the values before convolution.

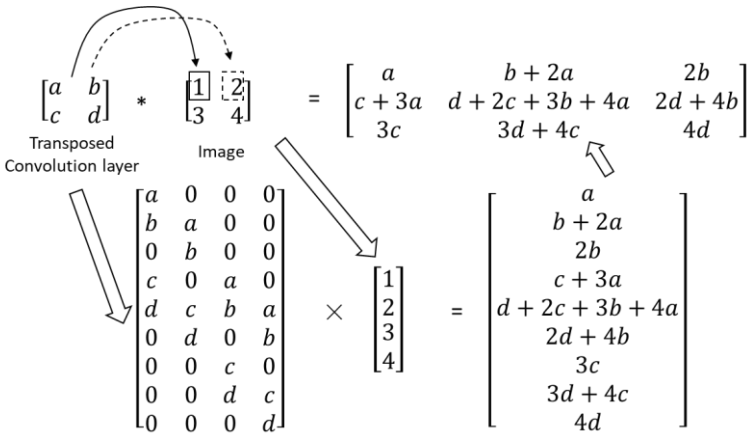


Figure 5 – Matrix multiplication form of a transposed convolution layer

3.1.6. Loss function

The output of DenseCrack is a feature map stating whether each pixel is crack or not. The design of loss function is the key for neural network training. It measures the error between predicted pixel label and the ground truth. The training process is in fact optimizing the weights of DenseCrack in order to minimize the loss function. In DenseCrack, a softmax function is used as the last layer [41]. The formula of the softmax layer is presented in equation (1) below.

$$q_0(i,j)=\frac{e^{x_0(i,j)}}{e^{x_0(i,j)}+e^{x_1(i,j)}},\ q_1(i,j)=\frac{e^{x_1(i,j)}}{e^{x_0(i,j)}+e^{x_1(i,j)}} \tag{1}$$

in which $q_0(i,j)$ and $q_1(i,j)$ stand for the probability of non-crack and crack; and $x_0(i,j)$ and $x_1(i,j)$ represent the input to softmax layer, i.e., output of second last layer at pixel (i,j), where x_0 is for non-crack and x_1 is for crack. It is noted that the sum of $q_0(i,j)$and $q_1(i,j)$ is 1, which satisfies the requirement of probability. The label with larger $q(i,j)$ would govern the prediction. Cross-entropy loss, as shown in equation (2), is calculated among the probabilities from all the pixels and is the object function of the training process.

$$Loss=\sum_{i,j\in\text{image}}\left[-p_0(i,j)\log q_0(i,j)-p_1(i,j)\log q_1(i,j)\right] \tag{2}$$

3.2. Depth first search based algorithm

The proposed neural network predicts each pixel interpedently but does not consider the property of continuity in cracks. This phenomenon was also observed by other researchers.[47] In this paper, we overcome this issue by considering the fact that cracks are connected areas with a large number of pixels while noise usually has much fewer number of pixels. In this algorithm, we convert the binary mask as a graph with 1 (crack) as nodes. There is an edge between any two nodes that are adjacent in the image. This process is illustrated in Figure 6~~Figure 6~~.

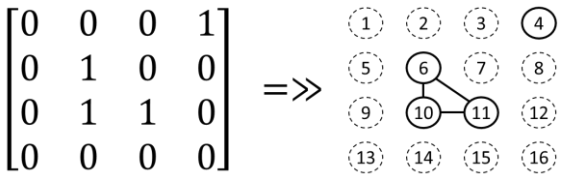
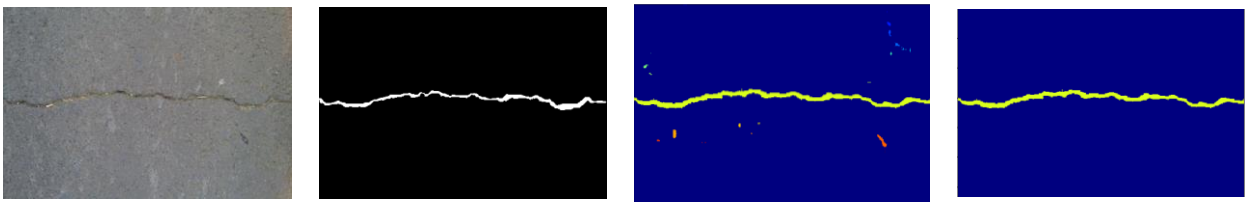


Figure 6 – Converting a binary mask to a graph

The problem is simplified to finding all the connected components in a graph and removing those with a small number of pixels. Therefore, a depth first search (DFS)[48] based algorithm is implemented on the binary mask to find connected components and label them accordingly. The reason we use DFS algorithm to find connected components because it is more memory efficient than other algorithms such as breadth-first search (BFS). The steps of the algorithm are summarized as below:

1) Apply morphological operations, i.e. dilation and erosion, to fill the small gaps in cracks.
2) Check the first pixel in the binary mask. If the pixel is assigned as crack by the neural network or is not labeled, then assign current label to this pixel, add this pixel to a new queue and go to step 3. If the pixel is assigned as non-crack by neural network or is already labelled, then repeat step 2 for the next pixel.
3) Pop out the pixel from the queue, and search all its neighbors. If the neighbor is crack and is not labeled, then assign current label to this neighbor, add this pixel to current queue. Repeat step 3 until the current queue becomes empty.
4) Increase the label number by 1 and go to step 2.
5) When all the pixels are traversed, count the number of pixels in each component and set all the pixels to non-crack in the components where the number of pixels is below the threshold.



(a) original image (b) ground truth (c) prediction (before thresholding) (d) prediction (after thresholding)

Figure 7 – Find connected components and apply thresholding

Figure 7~~Figure 7~~ presents the processing of a sample image with crack. In Figure 7~~Figure 7~~(c), DFS algorithm is applied to the raw output of the neural network. Each connected component is marked with a color. Comparing with Figure 7~~Figure 7~~(a), we know these small connected components come from stains or small defects on the surface. As shown in Figure 7~~Figure 7~~(d), after thresholding, the crack stands out with clean background.

4. Experimental Results and Discussions

4.1. Dataset and Training Process

To verify the proposed approach, a dataset called CFD dataset proposed by Shi et al.[8] is used in this paper so that a direct comparison among methods could be conducted. The dataset consists of 118 images

taken of road surfaces in Beijing, China. The images were taken by iPhone 5 with exposure time of 1/134 sec, focus of 4 mm and aperture of f/2.4. The resolution of the images is 480 by 320 pixels. The reason we choose this dataset for our studies is that this dataset is complete and is labelled with high quality at pixel level. The images in this dataset contains noise like shadows or stains. Also, since this dataset is used by a few other studies, the performance of our proposed network can be directly compared with the results from those studies.

Following the procedure in the paper[8], the total dataset is randomly split to 60% and 40% for training and test sets. After rounding up, 70 images are used for training and 48 images are for validation. To be more specific, each image is further split into 128 by 128 square patches with stride of 16. Therefore, each image corresponds to 299 small patches. To augment the data for training, each patch in training set is flipped horizontally and vertically once. Therefore, in total, there are 299×3×70=62790 patches for training.

For inference, i.e. detection of the crack, the small patches with the same size (128×128) are fed to the DenseCrack for pixel-level prediction. Since the stride is smaller than the size of patches, the mask for cracks in final image should be generated by voting from all the patches. For simplicity, the pixel is considered as crack as long as it is identified as crack in one patch.

The neural networks are trained on a PC with Intel 8700k CPU, 32GB memory and Nvidia Titan V GPU. Considering the memory limit of 11GB on GPU, the batch size of the training is set to 16. In this paper, patches instead of whole images are used for training and inference mainly due to the limited amount of available data. If image training is utilized, the weights of the neural network can be updated only once for each batch, and the information is not used efficiently.

4.2. Performance Evaluation

Three metrics are selected for the performance evaluation of the models, i.e., precision, recall and F1 score. The definition of these three metrics is given in equations (3), (4) and (5).

$$\text{precision} = \frac{TP}{TP + FP} \tag{3}$$

$$\text{recall} = \frac{TP}{TP + FN} \tag{4}$$

$$\text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{5}$$

In above equations, *TP* stands for the number of true positive pixels, *FP* is the number of false positive pixels, and *FN* is for the number of false negative pixels. In order to have a direct comparison with the method proposed in Shi et al.[8], true positives, false positives and false negatives follow their definition, which is re-phrased herein for readers' convenience. If a pixel is identified as crack, and it is within 5-pixel range of a ground truth crack pixel, this pixel can be considered as true positive. If a pixel is identified as crack, and it is outside of 5-pixel range of a ground truth crack pixel, this pixel is a false positive. If a pixel is identified as non-crack, but this pixel is in fact a crack pixel according to ground truth, this pixel is considered a false negative.

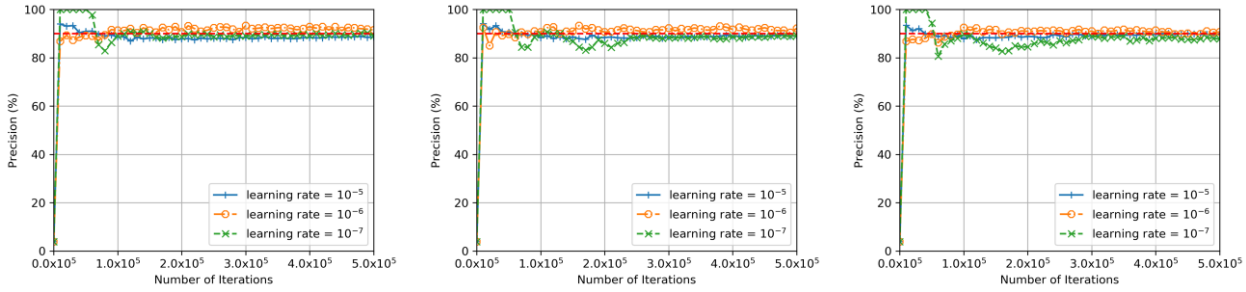
Following the definitions above, three variations of the DenseCrack, i.e. DenseCrack121, DenseCrack169 and DenseCrack201, are trained on the dataset described in last section. The "N" following each variation means the model is trained from scratch. At every 10000 iterations, the trained models are

evaluated on the reserved test set. In Figure 8~~Figure 8~~, the precision, recall and F1 score for test set are plotted against number of iterations for all three models. Three different learning rates are used for each model.

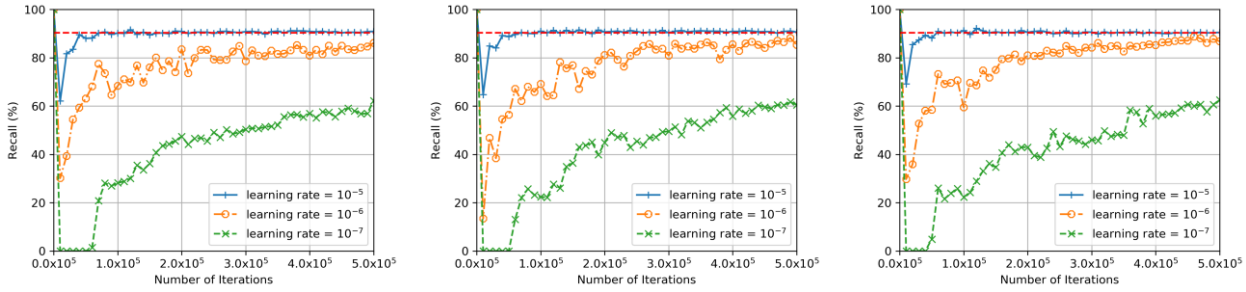
As shown in Figure 8~~Figure 8~~, precision and F1 score generally increase as the number of iterations increases. Recalls for all three models are 100% initially, and then drop to very low level before increase. This is because the models predict all pixels to label "crack" initially, and there are no false negatives, which leads to the recall of 100%. The shallower neural network is easier to train with low learning rate than deeper neural network because it is it has fewer parameters. It should be noted that when the learning rate is 10^{-7}, the recall drops to 0 before increase. This is a phenomenon called "all back" issue[49] where the neural network simply predicts every pixel as non-crack due to the unbalance of crack/non-crack labels in the dataset. The performance of the models becomes better when the learning rate increases. It should be noted that we investigated other learning rates as well, but higher learning rate does not give better performance beyond 10^{-5}. Therefore, considering the stability of the training process, 10^{-5} is chosen as the suitable learning rate for further analysis.

For each metric, the value from the model with the highest F1 score is added as a red dash line on the plots as a reference. The DenseCrack121-N can reach precision, recall and F1 score of 90.38%, 89.52% and 89.95% while DenseCrack169-N can reach 89.70%, 90.43% and 90.07%. The three metrics for DenseCrack201-N are 89.98%, 90.31% and 90.15%. Comparing all three variations, all precision, recall and F1 score converge to almost the same level. DenseCrack201-N gives the best performance in terms of F1 score, but the advantage is not very significant.

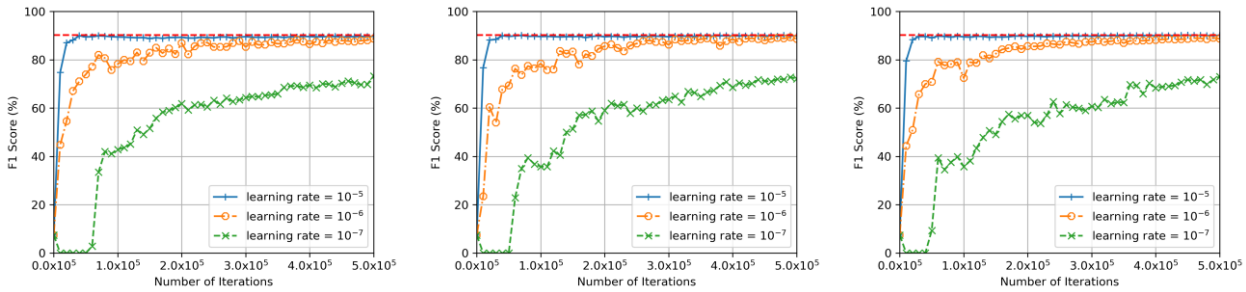
The best precision, recall and F1 score for each of the three variations are plotted against the number of parameters in Figure 11~~Figure 11~~ along with results from residual neural networks (ResNet).[43] The ResNet models follow the same procedure, including learning rate and postprocessing algorithm, as DenseCrack except that they use ResNet as backbone model instead of DenseNet. In Figure 9~~Figure 9~~, it is seen that DenseCrack outperforms ResNet based models with much fewer parameters. Even the shallowest DenseCrack121-N with only about 1/3 parameters as ResNet152 is 2% better than it.



(a) precision of DenseCrack121-N (b) precision of DenseCrack169-N (c) precision of DenseCrack201-N

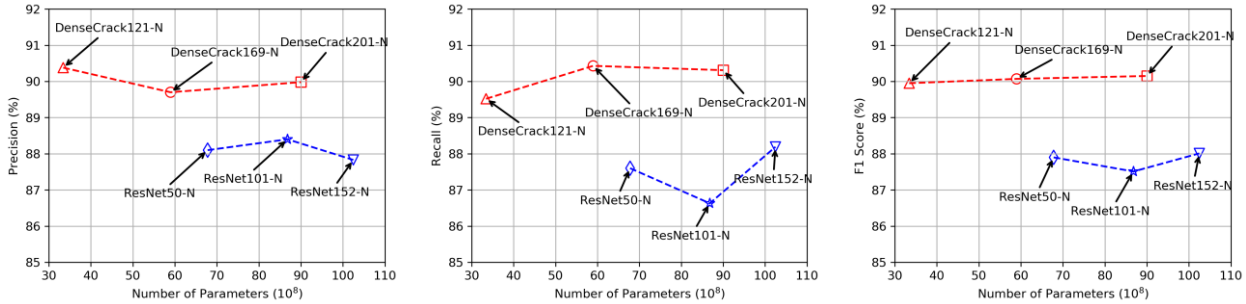


(d) recall of DenseCrack121-N (e) recall of DenseCrack169-N (f) recall of DenseCrack201-N



(g) F1 score of DenseCrack121-N (h) F1 score of DenseCrack169-N (i) F1 score of DenseCrack169-N

367 Figure 8 – Performance of DenseCrack121-N, DenseCrack169-N and DenseCrack201-N on test set



(a) precision (b) recall (c) F1 score

368 Figure 9 – Comparison of best performance for three different variations and ResNet

4.3. Influence of Feature Fusion Methods

In previous section, the final feature map for prediction is integrated from three different layers, i.e, outputs of Dense2, Dense3 and Dense4. Transposed convolution layers are applied to upsample from lower resolution to higher resolution features. In this section, the performance of DenseCrack201-N with different feature fusion methods is investigated. As can be seen in Figure 10~~Figure 10~~, the outputs of Dense2, Dense3 and Dense4 have dimensions of 16×16×512, 8×8×1792 and 4×4×1920, respectively. In the first method, which is termed as 32× feature fusion, a transposed convolution layer is directly applied to the output of Dense4 (4×4×1920) to do a 32-time upsampling and yield a feature map of 128×128×2. In the second method (16× feature fusion), the output of Dense4 is first upsampled by transposed convolution layer from 4×4×1920 to 8×8×1792, and is then added to the output of Dense3. Finally, the fused features are upsampled 16 times again from 8×8×1792 to 128×128×2. In the third method, outputs of Dense4 and Dense3 are fused first as in the second method, and are then upsampled the second time from 8×8×1792 to 16×16×512. Afterwards, the resulting features are added to the output of Dense2. The final feature map is obtained by upsampling the fused features 8 times from 16×16×512 to 128×128×2. The third method is called 8× feature fusion.

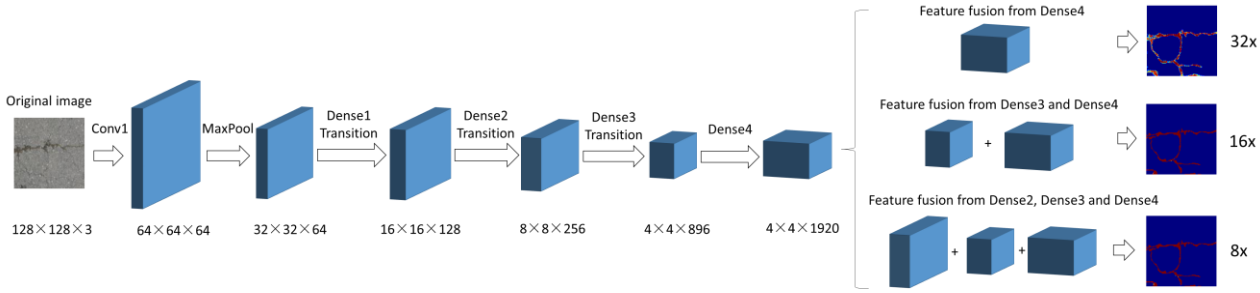


Figure 10 – Three different methods for feature fusion

The generated feature maps from all three feature fusion methods about some sample test images in CFD dataset are presented in Figure 11~~Figure 11~~. It should be noted that the feature maps are processed by softmax layer, so the color of a pixel in feature maps represents the probability of crack the neural networks assign to that pixel. The red color stands for 100%, while blue is for 0%. From Figure 11~~Figure 11~~, it is seen that all three methods could identify the location of cracks successfully. However, the 32× feature fusion method gives low confidence at the boundaries of the cracks and also gives some false positive predictions. 16× and 8× feature fusion methods are both better than 32× method. These results aligned with our expectations because 32× method only takes advantage of the last layer, and the information regarding details in previous layers are discarded.

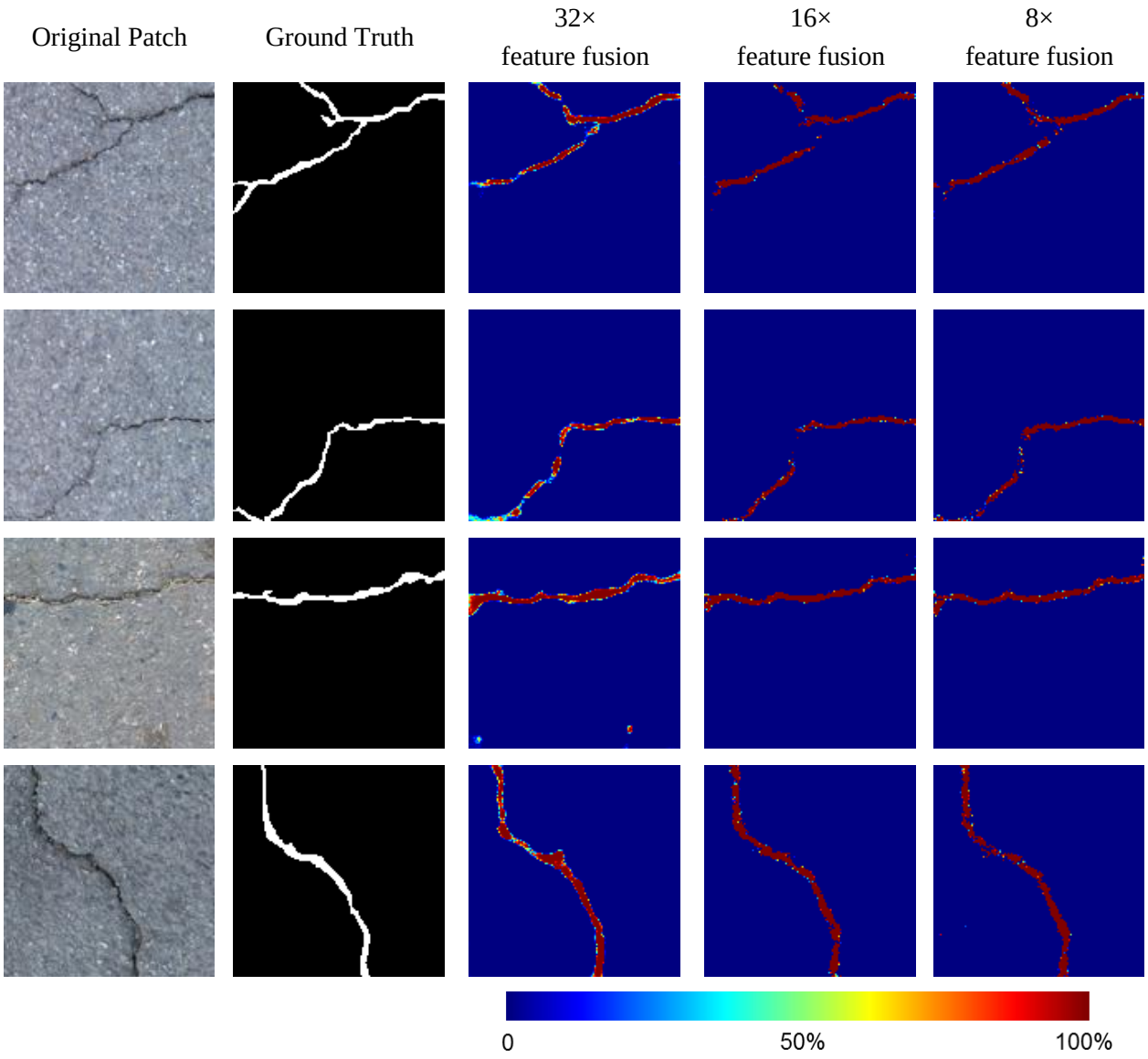
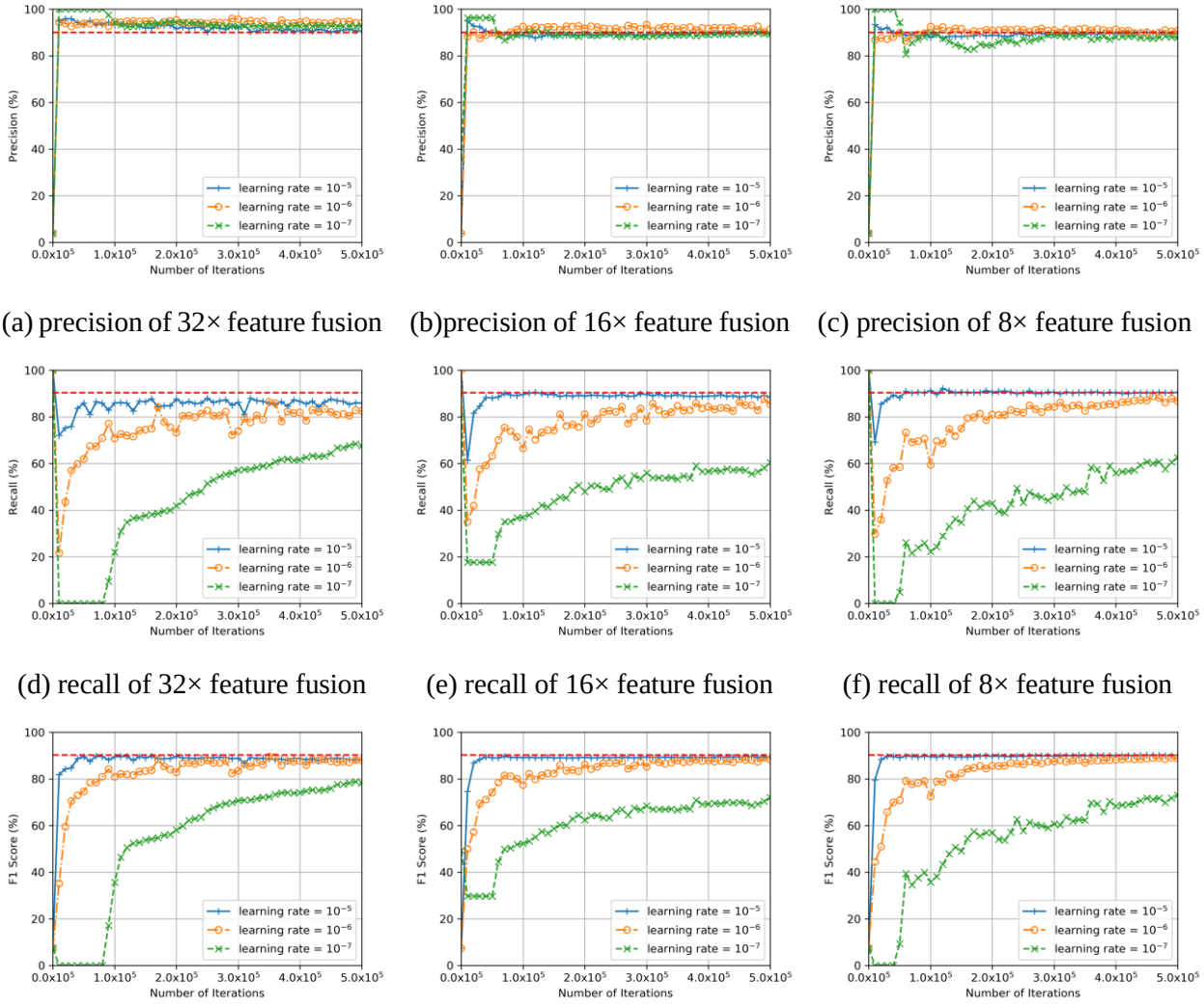


Figure 11 – Generated feature maps from different feature fusion methods

The precisions, recalls and F1 scores for all 3 feature fusion methods against number of iterations are plotted in Figure 12~~Figure 12~~. Similar to Figure 8~~Figure 8~~, three different learning rates are used. Comparing Figure 12~~Figure 12~~(a), (b) and (c), we can see that the recall and F1 score reach the highest when 8× feature fusion method is used. Also, 16× feature fusion method gives better performance than 32× feature fusion method in terms of F1 score and recall. It is noted that the model 32× method converge to higher precision faster with low learning rate. This is because there are fewer parameters to train with 32× method. The highest F1 scores for all 3 feature fusion methods along with their corresponding precisions and recalls are summarized in Table 1~~Table 1~~. It is concluded that among three feature fusion methods 8× method can achieve the best performance.



(a) precision of 32× feature fusion (b)precision of 16× feature fusion (c) precision of 8× feature fusion

(d) recall of 32× feature fusion (e) recall of 16× feature fusion (f) recall of 8× feature fusion

(g) F1 score of 32× feature fusion (h) F1 score of 16× feature fusion (i) F1 score of 8× feature fusion

Figure 12 – Performance of 32×, 16× and 8× feature fusion methods on test set

Table 1 – Comparison of performance for three different feature fusion methods

Feature fusion method	Precision	Recall	F1 score	Number of parameters	Computational cost (inference)
32×	91.80%	87.70%	89.70%	20,276,074	0.86 sec/image
16×	89.30%	90.00%	89.65%	34,956,650	1.27 sec/image
8×	89.98%	90.31%	90.15%	90,008,682	1.66 sec/image

4.4. Transfer Learning

In previous sections, the deep neural network models are all trained from scratch. It is well acknowledged that that transfer learning, in which the model is trained based on the weights of the model previously trained on other tasks[50], could boost the performance. In this section, the first 201 layers of DenseCrack201 is pretrained on a well-known dataset called ImageNet[51], which focuses on the detection of

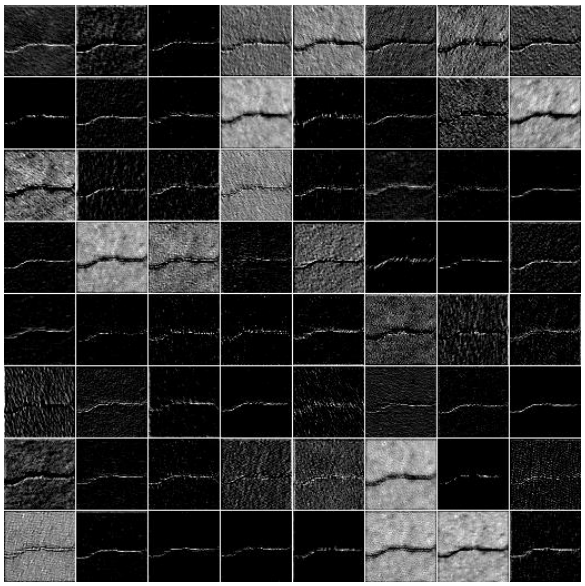
common objects such as dogs, people or cars. The transposed convolution layers are initialized randomly. It should be noted that there is no category for cracks in ImageNet dataset.

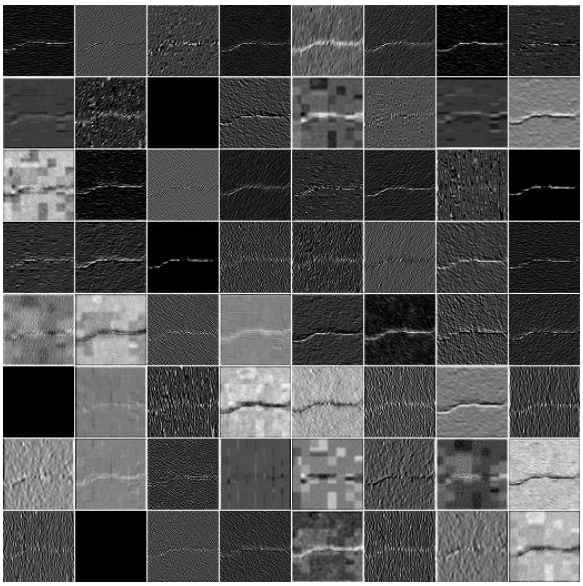
In the following sections, we refer the model using 8× feature fusion method without pretraining as DenseCrack201-N, and the model with pretraining as DenseCrack201-P. After training on the pretrained weights, the DenseCrack201-P model can reach a precision of 92.02%, a recall of 91.13% and a F1 score of 91.58% which are all higher than DenseCrack201-N. It is shown that transfer learning could help with the performance of our model even though our task is very different than training of ImageNet. In Figure 13~~Figure 13~~, the outputs of Conv1, Dense1 and Dense2 layers are plotted as grayscale images for visualization purpose. Even though the predictions for the model with and without pretraining are very similar as seen in Figure 13~~Figure 13~~ (b) and (c), Figure 13~~Figure 13~~ (d) through (i) show that the intermediate outputs are quite different. In the outputs of DenseCrack201-N, almost all the channels are activated due to the crack. However, for DenseCrack201-P, the number of activated pixels is much smaller than the one without pretraining, and the appearance of the output are different for different channels. Looking at Figure 14~~Figure 14~~, we plot the weights of the first layer for both models. We can see that the weights for DenseCrack201-N converge to the values which do not follow any patterns, but DenseCrack201-P's weights do have specific patterns. In fact, these weights in DenseCrack201-P serve as simple filters such as edge detectors or color detectors for feature extraction. Therefore, it is believed that DenseCrack201-N and DenseCrack201-P achieve two difference convergences, where DenseCrack201-P has better performance and is expected to have better generalization.

(a) original image

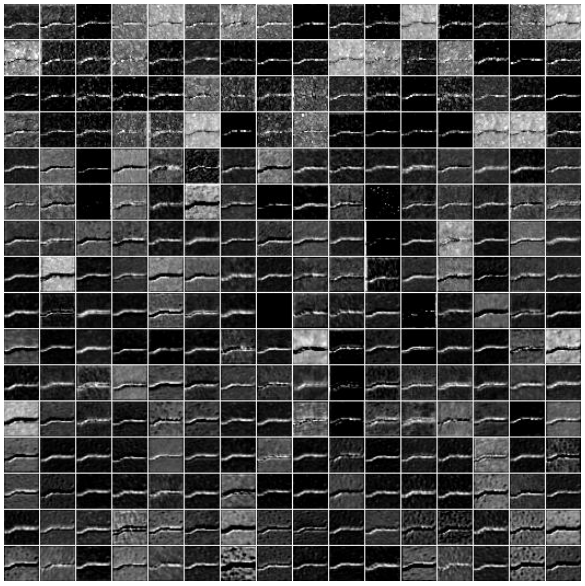
(b) prediction (without pretraining)

(c) prediction (with pretraining)

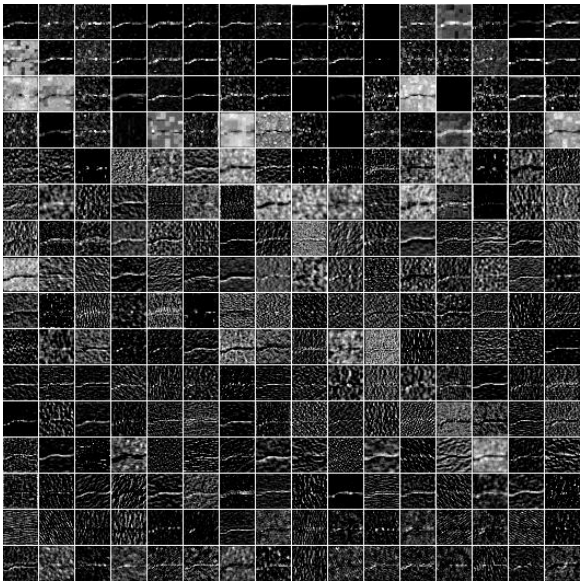




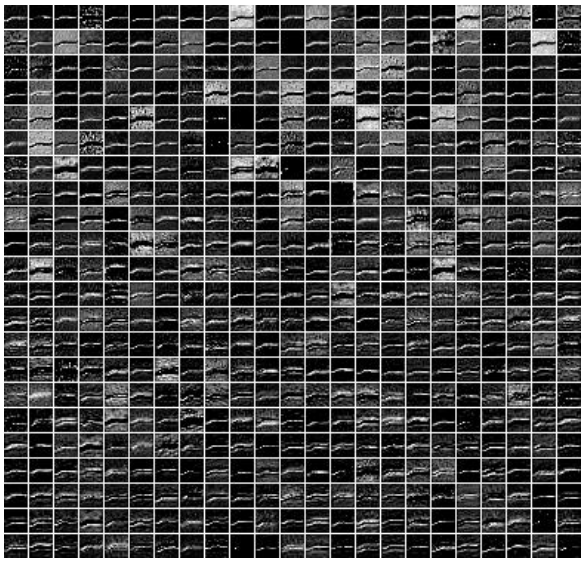
(d) Relu-activated output of Conv1
(without pretraining)



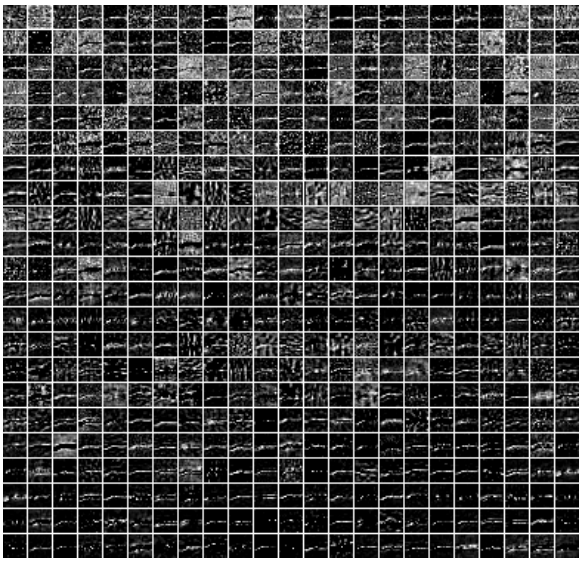
(e) Relu-activated output of Conv1
(with pretraining)



(f) integrated output of Dense1
(without pretraining)



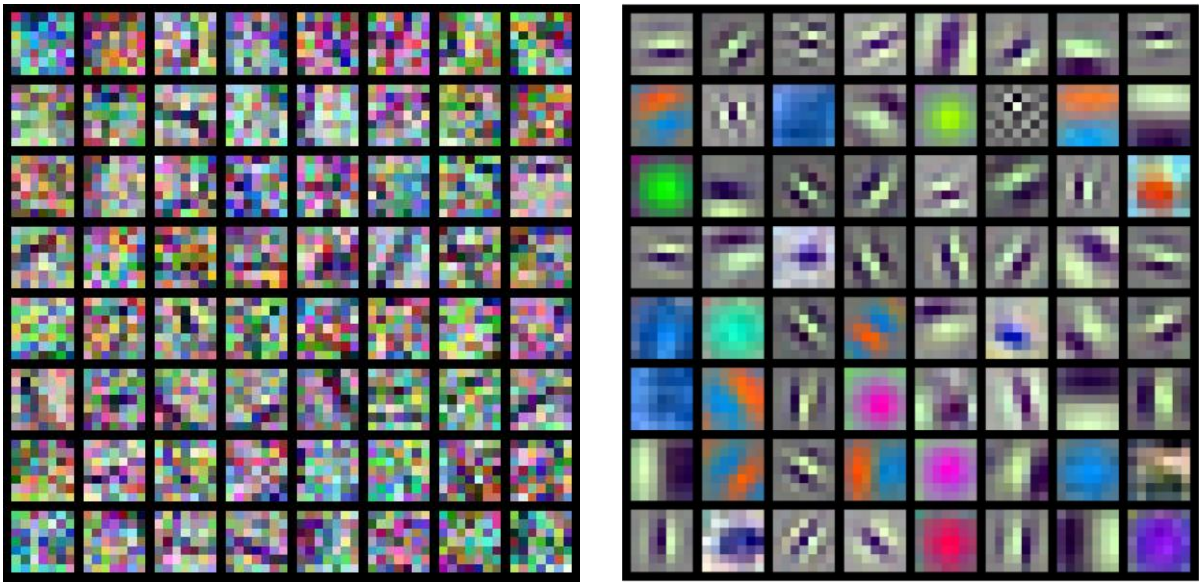
(g) integrated output of Dense1
(with pretraining)



(h) integrated output of Dense2
(without pretraining)

(i) integrated output of Dense2
(with pretraining)

Figure 13 – Visualization of selected layers on DenseCrack201-N and DenseCrack201-P

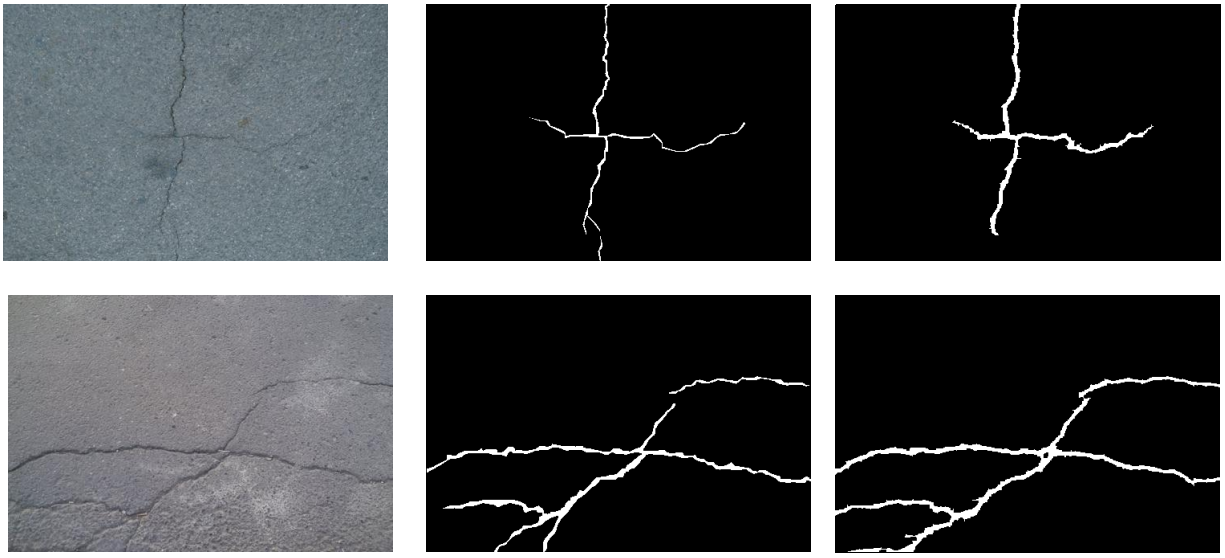


(a) DenseCrack201-N (b) DenseCrack201-P

Figure 14 – Visualization of weights for the first layer on DenseCrack201-N and DenseCrack201-P

In Figure 15~~Figure 15~~, some challenging images with a large amount of noise are presented together with prediction generated by DenseCrack201-P. It is seen that the model can do the crack segmentation at pixel level very accurately even with complex background. Especially for the last two sample images, we can see that the proposed method can distinguish cracks from other high frequency contents such as stains or edges of white paint. Interestingly, for the last sample image, DenseCrack201-P even successfully predicts the crack that was not ~~even~~correctly annotated in original dataset.

Original Image Ground Truth DenseCrack201-P



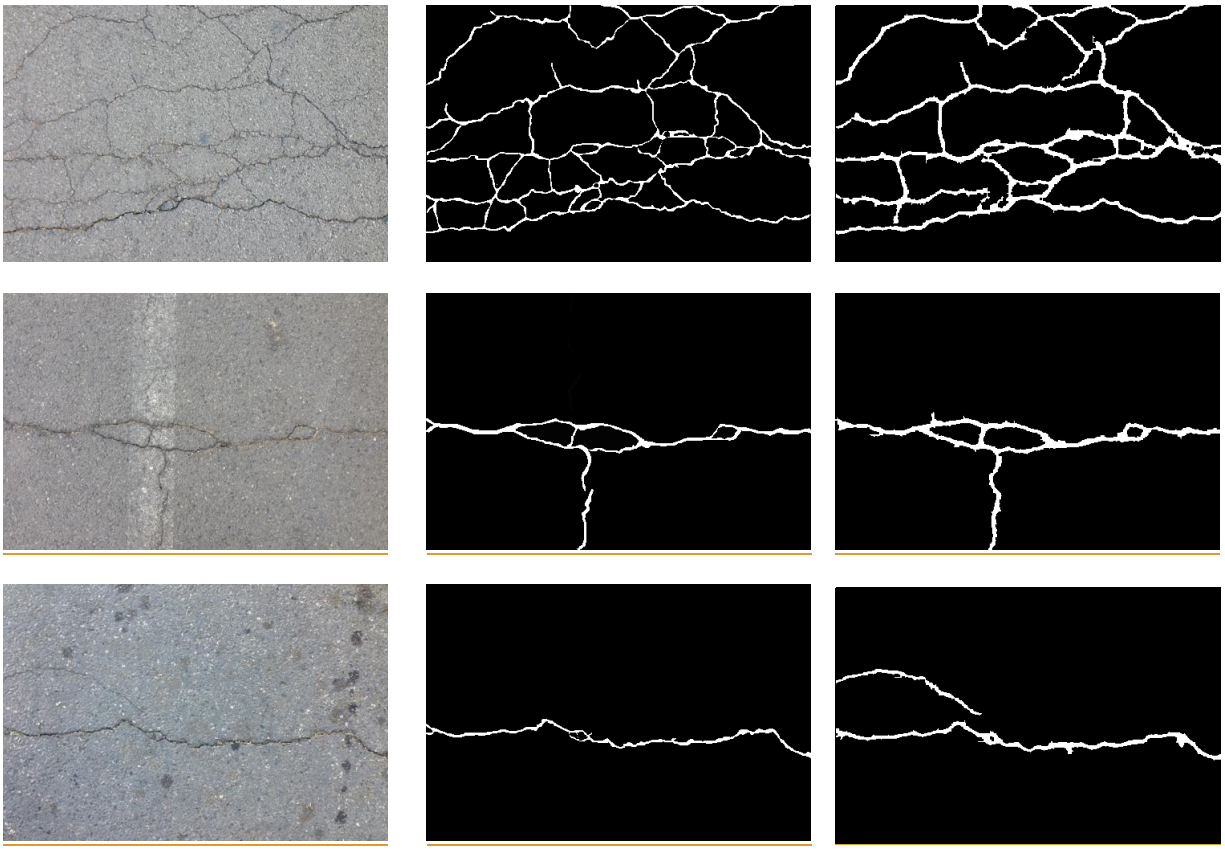


Figure 15 – Results for some challenging images predicted by DenseCrack201-P

4.5. Comparison with Other Methods

In this section, we compare the proposed method with other available methods on the same dataset. Among the available methods, Canny detector is the basic edge detection method. Free form anisotropy (FFA), which considers brightness and connectivity at the same time, was proposed by Nguyen et al. in 2011.[52] CrackTree based on minimum spanning tree was developed by Zou et al. in 2012.[53] CrackForest using random structured forests was proposed by Shi et al. in 2016.[8] It was also the paper which released the CFD dataset. Multi-scale fusion crack detection (MFCD) algorithm which is based on unsupervised learning was proposed by Li et al. in 2018.[54] Fei et al. developed a deep learning based algorithm called CrackNet-V in 2019.[55] In all these methods, CrackNet-V is deep learning based method while others are not. All these methods are tested under the same condition as our proposed method on 60%/40% training/test split on CFD data set. Overall, our proposed method is the only method yielding scores above 90% for all metrics. In terms of F1 score, both DenseCrack201-N and DenseCrack201-P give better performance than all other methods. Specifically, DenseCrack201-P outperforms the CrackForest by 5.87% and the state-of-the-art deep learning-based algorithm CrackNet-V by 2.40%. It should be noted that DenseCrack201-P has similar precision to CrackNet-V but much higher recall.

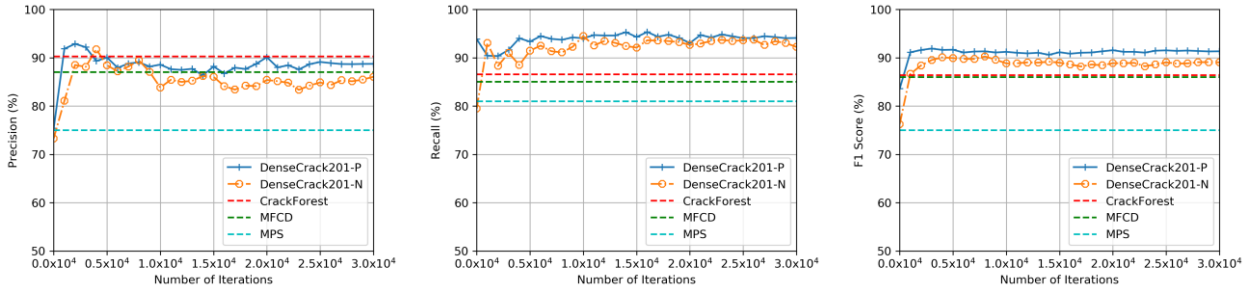
Table 2 – Comparison of performance for different methods

Method	Precision	Recall	F1 Score
Other methods			
Canny[8]	12.23%	22.15%	15.76%
FFA[54]	78.56%	68.43%	73.15%
CrackTree[8]	73.22%	76.45%	70.80%
CrackForest[8]	82.28%	89.44%	85.71%
MFCD[54]	89.90%	89.47%	88.04%
CrackNet-V[55]	**92.58%**	86.03%	89.18%
Proposed methods			
DenseCrack201-P	92.02%	**91.13%**	**91.58%**
DenseCrack201-N	89.98%	90.31%	90.15%

4.6. Generalization on a Different Dataset

The proposed method is also tested on a different dataset called AigleRN, which contains 38 labelled images on French pavement images.[56] The images have two different resolutions, 911×462 and 311×462 pixels. In our implementation, this dataset is also randomly split following 60%/40% rule for training and testing. Since the way this dataset labels the cracks is very different than CFD, the proposed methods, DenseCrack201-N and DenseCrack201-P, which were already trained on CFD dataset, are further trained on AigleRN. The performance of them is compared with CrackForest[8], MFCD[54] and minimum path selection (MPS)[56]. It should be noted that CrackForest follows the 60%/40% rule, but MFCD and MPS were applied on the whole dataset because they do not require training.

From Figure 16~~Figure 16~~, we can see that the precision of DenseCrack201-P converges to 92.17% within 30,000 iterations while DenseCrack201-N converges to 89.45% which is 2.72% lower than pretrained version. Both of them are slightly lower than CrackForest. In terms of recall, both DenseCrack201-P and DenseCrack201-N are significantly higher than other methods. They can reach 91.61 and 91.09% respectively when converging. Regarding the F1 score which comprehensively considers precision and recall, both DenseCrack201-P and DenseCrack201-N can beat other methods. Specifically, DenseCrack201-P has 1.63% better F1 score than DenseCrack201-N. This shows that the proposed method has very good capability for generalization on other dataset, and pretraining on a large dataset could further help the generalization.



(a) precision (b) Recall (c) F1 score

Figure 16 – Performance comparison on AigleRN

5. Conclusions

In this paper, a pixel-level crack segmentation method including a novel densely connected deep neural network architecture and a DFS based thresholding is proposed. Testing on CFD dataset, the dense block is compared with the residual block. The influence of feature fusion methods and transfer learning is studied. The generalization of this method on another dataset is also presented. From our studies, the following main conclusion are obtained:

1) The proposed method with pretraining can reach a precision of 92.02%, a recall of 91.13% and a F1 score of 91.58% on CFD dataset, which is much higher than the original study which released CFD dataset and also higher than other state-of-the-art methods in terms of recall and F1 score.
2) This study shows feature fusion from multiple layers could provide better performance than a single layer. More layers are better for the prediction.
3) Transfer learning can boost the performance of the model on CFD dataset, and helps generalizing on another dataset.

The paper focuses on developing a novel deep learning-based method for crack segmentation, which is tested on publicly available datasets. However, the limitations of these datasets include: 1) the datasets are specifically designed to only include the pavement surfaces within a certain distance; 2) the datasets do not explicitly include challenging images with varying illumination conditions or challenging noise; 3) only one smartphone is used all the time for data collection, and all the images have the same resolution.

In the future, the proposed method and its variations will be tested on more challenging datasets collected by our research team. The new dataset will be taken from different moving cars or drones at different weather conditions. In future, more critical cases will be collected for training and testing so that the method will be more robust in complex situations. We will also explicitly study the effect of illumination conditions or noise on the proposed methods.

Declaration of conflicting interests

The authors declare no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

1. Félio G. Informing the Future: The Canadian Infrastructure Report Card. *VICTORIA* 2016.
2. Zhang L, Yang F, Zhang YD, et al. Road crack detection using deep convolutional neural network. In: *Image Processing (ICIP), 2016 IEEE International Conference on* 2016, pp.3708-3712. IEEE.
3. Kim J-T and Stubbs N. Crack detection in beam-type structures using frequency data. *Journal of Sound and Vibration* 2003; 259: 145-160.
4. Quek S-T, Wang Q, Zhang L, et al. Sensitivity analysis of crack detection in beams by wavelet technique. *International journal of mechanical sciences* 2001; 43: 2899-2910.
5. Shan Q and Dewhurst R. Surface-breaking fatigue crack detection using laser ultrasound. *Applied Physics Letters* 1993; 62: 2649-2651.

6. Glushkov E, Glushkova N, Ekhlakov A, et al. An analytically based computer model for surface measurements in ultrasonic crack detection. *Wave Motion* 2006; 43: 458-473.
7. Owolabi G, Swamidas A and Seshadri R. Crack detection in beams using changes in frequencies and amplitudes of frequency response functions. *Journal of sound and vibration* 2003; 265: 1-22.
8. Shi Y, Cui L, Qi Z, et al. Automatic road crack detection using random structured forests. *IEEE Transactions on Intelligent Transportation Systems* 2016; 17: 3434-3445.
9. Jahanshahi MR, Kelly JS, Masri SF, et al. A survey and evaluation of promising approaches for automatic image-based defect detection of bridge structures. *Structure and Infrastructure Engineering* 2009; 5: 455-486.
10. Chen Y, Logan P, Avitabile P, et al. Non-Model Based Expansion from Limited Points to an Augmented Set of Points Using Chebyshev Polynomials. *Experimental Techniques* 2019: 1-23.
11. LeBlanc B, Niezrecki C, Avitabile P, et al. Damage detection and full surface characterization of a wind turbine blade using three-dimensional digital image correlation. *Structural Health Monitoring* 2013; 12: 430-439.
12. He K, Gkioxari G, Dollár P, et al. Mask r-cnn. In: *Computer Vision (ICCV), 2017 IEEE International Conference on* 2017, pp.2980-2988. IEEE.
13. Cha YJ, Choi W and Büyüköztürk O. Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering* 2017; 32: 361-378.
14. Cha YJ, Choi W, Suh G, et al. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Computer-Aided Civil and Infrastructure Engineering* 2018; 33: 731-747.
15. Chen F-C and Jahanshahi MR. NB-CNN: deep learning-based crack detection using convolutional neural network and naive Bayes data fusion. *IEEE Transactions on Industrial Electronics* 2018; 65: 4392-4400.
16. Maeda H, Sekimoto Y, Seto T, et al. Road damage detection and classification using deep neural networks with smartphone images. *Computer-Aided Civil and Infrastructure Engineering* 2018; 33: 1127-1141.
17. Mandal V, Uong L and Adu-Gyamfi Y. Automated Road Crack Detection Using Deep Convolutional Neural Networks. In: *2018 IEEE International Conference on Big Data (Big Data)* 2018, pp.5212-5215. IEEE.
18. Schmugge SJ, Rice L, Nguyen NR, et al. Detection of cracks in nuclear power plant using spatial-temporal grouping of local patches. In: *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)* 2016, pp.1-7. IEEE.
19. Xue Y and Li Y. A fast detection method via region-based fully convolutional neural networks for shield tunnel lining defects. *Computer-Aided Civil and Infrastructure Engineering* 2018; 33: 638-654.
20. Bang S, Park S, Kim H, et al. Encoder–decoder network for pixel-level road crack detection in black-box images. *Computer-Aided Civil and Infrastructure Engineering* 2019.
21. Dung CV and Anh LD. Autonomous concrete crack detection using deep fully convolutional neural network. *Automation in Construction* 2019; 99: 52-58. DOI: https://doi.org/10.1016/j.autcon.2018.11.028.
22. Abdel-Qader I, Abudayyeh O and Kelly ME. Analysis of edge-detection techniques for crack identification in bridges. *Journal of Computing in Civil Engineering* 2003; 17: 255-263.
23. Sinha SK and Fieguth PW. Automated detection of cracks in buried concrete pipe images. *Automation in Construction* 2006; 15: 58-72.
24. Cha Y-J, You K and Choi W. Vision-based detection of loosened bolts using the Hough transform and support vector machines. *Automation in Construction* 2016; 71: 181-188.
25. Iyer S and Sinha SK. A robust approach for automatic detection and segmentation of cracks in underground pipeline images. *Image and Vision Computing* 2005; 23: 921-933.

26. Lecompte D, Vantomme J and Sol H. Crack detection in a concrete beam using two different camera techniques. *Structural Health Monitoring* 2006; 5: 59-68.
27. Jung HK and Park G. Rapid and non-invasive surface crack detection for pressed-panel products based on online image processing. *Structural Health Monitoring* 2019: 1475921718811157.
28. Fujita Y and Hamamoto Y. A robust automatic crack detection method from noisy concrete surfaces. *Machine Vision and Applications* 2011; 22: 245-254.
29. Adhikari R, Moselhi O and Bagchi A. Image-based retrieval of concrete crack properties for bridge inspection. *Automation in construction* 2014; 39: 180-194.
30. Turkan Y, Hong J, Laflamme S, et al. Adaptive wavelet neural network for terrestrial laser scanner-based crack detection. *Automation in Construction* 2018; 94: 191-202.
31. Protopapadakis E, Stentoumis C, Doulamis N, et al. AUTONOMOUS ROBOTIC INSPECTION IN TUNNELS. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* 2016; 3.
32. Kim H, Ahn E, Shin M, et al. Crack and noncrack classification from concrete surface images using machine learning. *Structural Health Monitoring* 2019; 18: 725-738.
33. Makantasis K, Protopapadakis E, Doulamis A, et al. Deep convolutional neural networks for efficient vision based tunnel inspection. In: *2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)* 2015, pp.335-342. IEEE.
34. Feng C, Liu M-Y, Kao C-C, et al. Deep active learning for civil infrastructure defect detection and classification. *Mitsubishi Electric Research Laboratories Available online: https://www merl com/publications/docs/TR2017-034 pdf (accessed on 6 June 2018)* 2017.
35. Protopapadakis E and Doulamis N. Image based approaches for tunnels' defects recognition via robotic inspectors. In: *International Symposium on Visual Computing* 2015, pp.706-716. Springer.
36. Zhang A, Wang KC, Li B, et al. Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network. *Computer-Aided Civil and Infrastructure Engineering* 2017; 32: 805-819.
37. Ni F, Zhang J and Chen Z. Pixel-level crack delineation in images with convolutional feature fusion. *Structural Control and Health Monitoring* 2019: e2286.
38. Liu Z, Cao Y, Wang Y, et al. Computer vision-based concrete crack detection using U-net fully convolutional networks. *Automation in Construction* 2019; 104: 129-139. DOI: 10.1016/j.autcon.2019.04.005.
39. LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 1998; 86: 2278-2324.
40. Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* 2016, pp.779-788.
41. Goodfellow I, Bengio Y, Courville A, et al. *Deep learning*. MIT press Cambridge, 2016.
42. Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* 2015, pp.1-9.
43. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* 2016, pp.770-778.
44. Hochreiter S, Bengio Y, Frasconi P, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. A field guide to dynamical recurrent neural networks. IEEE Press, 2001.
45. Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* 2017, pp.4700-4708.

46. Long J, Shelhamer E and Darrell T. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* 2015, pp.3431-3440.

47. Doulamis A, Doulamis N, Protopapadakis E, et al. Combined convolutional neural networks and fuzzy spectral clustering for real time crack detection in tunnels. In: *2018 25th IEEE International Conference on Image Processing (ICIP)* 2018, pp.4153-4157. IEEE.

48. Cormen TH, Leiserson CE, Rivest RL, et al. *Introduction to algorithms*. MIT press, 2009.

49. Yang X, Li H, Yu Y, et al. Automatic pixel-level crack detection and measurement using fully convolutional network. *Computer-Aided Civil and Infrastructure Engineering* 2018; 33: 1090-1109.

50. Pan SJ and Yang Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 2010; 22: 1345-1359.

51. Deng J, Dong W, Socher R, et al. Imagenet: A large-scale hierarchical image database. In: *Computer Vision and Pattern Recognition, 2009 CVPR 2009 IEEE Conference on* 2009, pp.248-255. Ieee.

52. Nguyen TS, Begot S, Duculty F, et al. Free-form anisotropy: A new method for crack detection on pavement surface images. In: *2011 18th IEEE International Conference on Image Processing* 2011, pp.1069-1072. IEEE.

53. Zou Q, Cao Y, Li Q, et al. CrackTree: Automatic crack detection from pavement images. *Pattern Recognition Letters* 2012; 33: 227-238.

54. Li H, Song D, Liu Y, et al. Automatic Pavement Crack Detection by Multi-Scale Image Fusion. *IEEE Transactions on Intelligent Transportation Systems* 2018: 1-12.

55. Fei Y, Wang KC, Zhang A, et al. Pixel-Level Cracking Detection on 3D Asphalt Pavement Images Through Deep-Learning-Based CrackNet-V. *IEEE Transactions on Intelligent Transportation Systems* 2019.

56. Amhaz R, Chambon S, Idier J, et al. Automatic crack detection on two-dimensional pavement images: An algorithm based on minimal path selection. *IEEE Transactions on Intelligent Transportation Systems* 2016; 17: 2718-2729.

List of Tables

List of Figures