

# Automated Pavement Crack Segmentation Using Fully Convolutional U-Net with a Pretrained ResNet-34 Encoder

Stephen L. H. Lau, Xin Wang, *Senior Member, IEEE*, Xu Yang, and Edwin K. P. Chong, *Fellow, IEEE*

**Abstract**—Automated pavement crack segmentation is a challenging task because of inherent irregular patterns and lighting conditions, in addition to the presence of noise in images. Conventional approaches require a substantial amount of feature engineering to differentiate crack regions from non-affected regions. In this paper, we propose a deep learning technique based on a convolutional neural network to perform segmentation tasks on pavement crack images. Our approach requires minimal feature engineering compared to other machine learning techniques. The proposed neural network architecture is a modified U-Net in which the encoder is replaced with a pretrained ResNet-34 network. To minimize the dice coefficient loss function, we optimize the parameters in the neural network by using an adaptive moment optimizer called AdamW. Additionally, we use a systematic method to find the optimum learning rate instead of doing parametric sweeps. We used a “one-cycle” training schedule based on cyclical learning rates to speed up the convergence. We evaluated the performance of our convolutional neural network on CFD, a pavement crack image dataset. Our method achieved an *F1* score of about 96%. This is the best performance among all other algorithms tested on this dataset, outperforming the previous best method by a 1.7% margin.

**Index Terms**—Convolutional neural network, deep learning, fully convolutional network, pavement crack segmentation, U-Net.

## I. INTRODUCTION

CRACK formation on pavements poses a safety hazard on road users. The main causes of pavement crack formation include traffic, moisture, and construction quality [1]. The quality of roads worsens with time owing to wear and tear. Continual traffic flow in urban areas exacerbates this problem. A study done in 2006 revealed that accidents due to road conditions in the United States alone cost \$217.5 billion [2]. The risk increases with road usage, and the consequence can be as serious as death. The maintenance of pavements is therefore a priority to ensure safety of road users. One way to improve pavement maintenance is by using computational algorithms to segment the pavement cracks from the background. If it achieves a reasonable accuracy, then it can usefully be deployed by pavement engineers to replace the traditional method of

visually inspecting the pavement cracks before repairing them, since computer algorithms are more efficient.

Image processing techniques such as thresholding [3], mathematical morphology [4], and edge detection [5] are often used to segment pavement cracks, but are very sensitive to noise in images and are unable to generalize the differentiation of a crack from its background. Methods such as local binary pattern operator and wavelet transforms are often applied to take geometry into account. They are, however, often unable to detect all forms of cracks. Kaul et al. [6] proposed an algorithm based on a minimal path method to predict open curves without knowledge of the endpoints or topology of the curves. However, the computational cost of this method is a major practical drawback. Shi et. al. [7] proposed a crack detection framework based on random forests, which heavily relies on feature extraction of an annotated image database.

Conventional segmentation methods based on manual techniques and filtering have been used for crack segmentation. Zhang [8] used a thresholding method for segmentation to smooth input images and detect cracks by calculating the area and perimeter of the roundness index. Salman et al. [9] proposed to distinguish cracks based on the Gabor filtering. Shan et al. [10] used the Canny-Zernike algorithm to obtain the image coordinates of a crack edge on the recovered coordinates of two stereo vision cameras. Iyer and Sinha [11] developed a statistical filter for feature extraction before detecting cracks by cleaning and linking. Talab et al. [12] used Sobel’s filter before using thresholding to separate the foreground from the background. Amhaz et al. [13] proposes an algorithm for crack segmentation based on the selection of minimal paths. That method takes both photometric and geometric characteristics into account and requires little a priori information.

Morphological-based algorithms have also been applied to crack segmentation. Morphological operations involve the collections of non-linear operations such as erosion, dilation, and watershed transform, associated with the shape or features in an image. Iyer and Sinha [11] designed a method to detect crack patterns using curvature evaluation and mathematical morphology. Merazi-Meksen et al. [14] also used this approach to extract relevant pixels corresponding to the presence of

Manuscript received July 24, 2019; revised ; accepted . Date of publication ; date of current version . The Associate Editor for this paper was Azim Eskandarian. (*Corresponding author: Xin Wang.*)

Stephen L. H. Lau and Xin Wang are with Department of Mechanical Engineering, Monash University Malaysia, 47500 Subang Jaya, Selangor, Malaysia (email: stephen.lau@monash.edu, wang.xin@monash.edu).

Xu Yang is with Department of Civil Engineering, Monash University, Clayton VIC 3800, Australia (email: xu.yang@monash.edu).

Edwin K. P. Chong is with the Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523, USA (email: EdwinChong@ColoState.Edu).

discontinuities, which are then characterized using a pattern recognition technique.

More sophisticated systems and algorithms have also been adopted. For instance, Yang et al. [15] combined the use of artificial neural networks, traditional crack depth prediction methods, and sub-pixel width measurements. Zou et al. [16] used a geodesic shadow removal algorithm to remove pavement shadows, followed by the tensor voting methods to build a crack probability map. Oliveira and Correia [17] designed a system using unsupervised training on a subset of the training dataset. Lins and Givigi [18] developed a system based machine-vision system in which a single camera processes a sequence of images for estimating crack dimensions.

Deep learning has become popular in computer vision. The method is able to learn the features required for classification, rather than manually extracting them [19]. Convolutional neural networks (CNNs) have also gained attention in the machine learning community after AlexNet won the 2012 ImageNet competition in image classification, significantly improving the previous top-5 accuracy score from about 74% to 84% [20]. Since then, CNNs have been used widely in image processing tasks such as classification [21, 22], object detection [23], and segmentation [24].

Fan et al. [25] proposed a method using a CNN for structural prediction as a multi-label problem. Small image patches are extracted from an image and each patch is labeled *positive* if the center pixel is a crack pixel, and *negative* otherwise. The prediction output corresponds to a 5×5 image. Despite yielding good results, the method suffers from a significantly long inference times per testing image. Zhang et al. [26] used a CNN for segmentation. Their approach involves a feature extractor to provide input to a second CNN. The same authors also used a recurrent neural network on 3D pavement images to predict the local paths with highest probability to form crack patterns [27]. These methods are not end-to-end networks and still require feature extraction. Yang et al. [28] proposed a novel architecture that involves feature pyramid and hierarchical boosting modules. The feature pyramid modules merge the feature maps from two successive CNN layers in the downsampling blocks, whereas the hierarchical boosting modules assign weights to easy and hard samples accordingly.

In this paper, we propose a deep learning algorithm for pavement crack segmentation, which we demonstrate to outperform other methods. Our deep learning algorithm reduces the need for manual feature extraction because it is able to learn the important features required to classify each pixel as a crack or not.

Section I of this paper explains the motivation of our work and provides a brief review of the previously deployed image processing methodologies and machine learning algorithms. Section II discusses the related work that we built on to develop our algorithm. This includes convolutional neural networks for segmentation and state-of-the-art optimization techniques. Section III introduces the dataset we used to train and test our neural network, as well as the methodology we used to pre-process it. In Section IV, we explain the architecture of our convolutional neural network, the loss function we adopted, and the various steps of the training procedure. In Section V, we provide prediction results on the validation dataset and the quantitative metrics we used to evaluate the results. In addition,

we also compare our results with other methodologies. Finally, Section VI and concludes the paper by summarizing our work while pointing out its limitations and some ways to improve it.

## II. RELATED WORK

### A. Residual Networks (ResNets)

He et al. [29] investigated if stacking more layers leads to a better neural network. They realized that the problem with increasing the number of layers is vanishing or exploding gradients. Residual mappings can be used to alleviate this issue. The resulting network architecture, ResNet, won several categories in the 2015 ImageNet competition as well as COCO segmentation tasks. This network architecture proves that deep neural networks of thousands of layers are trainable. The same network was also used by the winners of the DAWNBench competition [30] which prioritizes reducing training cost while obtaining a 94% test accuracy on CIFAR-10 and 93% top-5 accuracy scores on ImageNet, respectively. This shows the computational efficiency of ResNet while obtaining good accuracy in image recognition tasks.

### B. Network Architectures for Segmentation

Long et al. [24] proposed a network architecture called a fully convolutional network, which improves the performance of semantic segmentation tasks. The network is able to take an input of any size and produce an output of corresponding size. Ronneberger et al. [31] proposed a novel network architecture, U-Net, to improve segmentation performance on biomedical microscopy images. The main contribution of [31] is an architecture that connects the downsampling layers to the upsampling layers. The reason behind this connection is for the upsampling layers to directly obtain the features of the various downsampling layers to compensate for the loss in information during downsampling. For pavement crack segmentation tasks, several researchers, such as Jenkins et al. [32], Nguyen et al. [33], and Cheng et al. [34], also used a U-Net-based network architecture in their respective supervised learning algorithms.

### C. Systematic Learning Rate Finder and “One-Cycle” Training Schedule

Cyclical learning rate training schedule was introduced by Smith [35], subsequently updated in [36] with some additional suggestions on tuning the hyperparameters. The author claimed that super-convergence, a phenomenon in which a deep neural network is trained in orders of magnitude faster than conventional training methods, is possible to achieve by carefully setting the important hyperparameters such as the learning rate. This is achieved via a novel method to find the optimum learning rate, coupled with a training regime, based on cyclical learning rates, called a “one-cycle” training schedule. The proposed method allows for higher learning rates, and is therefore able to achieve a given test accuracy on CIFAR-10 and CIFAR-100 in a significantly fewer number of epochs.

### D. AdamW

Adaptive gradient optimizers have been introduced to improve upon stochastic gradient descent (SGD). One of the proposed optimizers is Adam [37], which was claimed to have

achieved significantly better results on MNIST and IMDB review databases. However, further experiments by Wilson et al. [38] discovered that Adam performed worse than a network optimized with SGD and momentum. Loshchilov and Hutter [39] then investigated the reason for this phenomenon. They realized that the equivalency of  $L_2$  regularization to weight decay, though applicable in SGD, does not apply to adaptive gradient optimization methods such as Adam. Hence, [39] proposed a method called AdamW in an attempt to decouple the weight decay from gradient-based parameter updates. From their experiments, they found that AdamW achieved better results than Adam on CIFAR-10 and ImageNet32×32. In addition, AdamW performed better than SGD with momentum, just as theoretically expected. Furthermore, they found that the weight decay is also decoupled from the learning rate. Hence, these two hyperparameter settings can be set independently without worrying if one affects the other.

### III. DATASET

The CFD dataset [7] was used as the dataset for our present study. The dataset consists of 118 images of pavement cracks, captured using a smartphone from vertically above the pavements, together with the manually labeled binary ground truths for each image. Each image has a resolution of 320 pixels (height) by 480 pixels (width). This dataset was used to train and test our proposed neural network.

We split the dataset into training and validation sets by a 60:40 ratio, resulting in 72 images in the training set and 46 images in the validation set. Fig. 1 shows example images from the training set. To produce more images, we extracted smaller images of size 128×128, 256×256, and 320×320 from each 320×480 image in both training set and validation set. This was done by cropping out the smaller images at a stride of 20 pixels in both horizontal and vertical axes. Our CNN was trained on

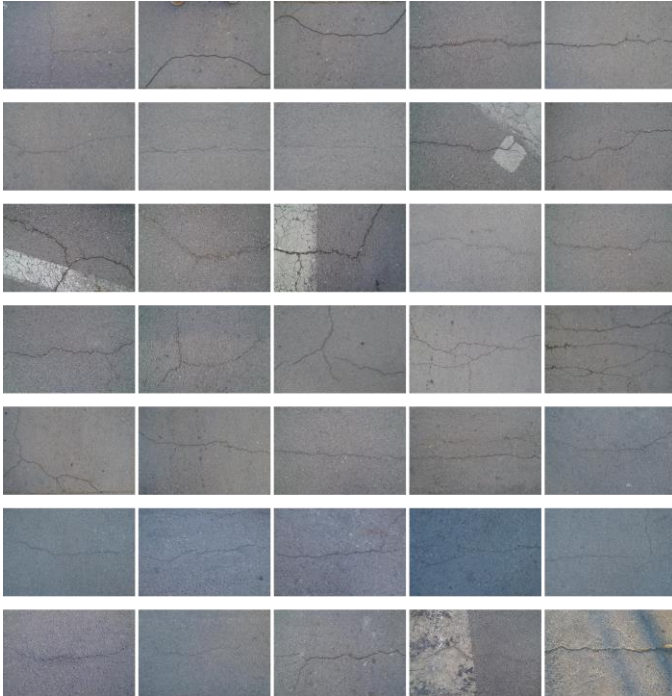


Fig. 1. Example images from the training set

these increasing image sizes. Cropping was performed identically on their respective ground truth images to ensure that each cropped training image matches their corresponding ground truth. Table I shows the distribution of the datasets for the various image sizes.

TABLE I  
DATASET DISTRIBUTION FOR VARIOUS IMAGE SIZES

Image size	Number of images	
	Training set	Validation set
128×128	12960	8280
256×256	3456	2208
320×320	1224	368

Data augmentations were performed to virtually increase the number of training examples. In each iteration of the neural network training, three types of augmentations were randomly performed on the training images:

- Random rotation between 0° to 360°. To deal with the issue of corners of rotated images, the pixels near the rectangular corners take the reflection of the borders of the rotated image.
- Random flip in horizontal and vertical axes.
- Random change in lighting. The image balance and contrast are randomly increased or decreased by a change of 0.05 respectively.

The same augmentations, except for lighting, were applied on the ground truth of each training image.

### IV. PROPOSED METHOD

The method we propose involves a supervised learning algorithm in which we optimally fit a function approximator  $f: \mathbf{x} \rightarrow \mathbf{y}$  using training data with the goal of mapping images  $\mathbf{x}$  to their respective labels  $\mathbf{y}$ . In deep learning, this function approximator takes the form of a parameterized neural network consisting of multiple layers of weights and biases. The training is done using a loss function  $L = g(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta})$ , which the neural network training seeks to minimize by tuning  $\boldsymbol{\theta}$ . This tuning is done with a gradient descent-based optimizer.

#### A. Network Architecture

The network architecture we propose is a U-Net [31] based architecture with a ResNet-34 [29] encoder, as shown in Fig. 2. (Our description assumes some background in convolutional neural networks.) The ResNet-34 encoder was pretrained on ImageNet. This fully convolutional network receives a three-channel (RGB) image as input and produces a one-channel (binary) output of the same size. The CNN begins with a convolutional layer with a kernel size of 7×7 and stride of 2 to downsample the input by half. This is followed by a batch normalization (BN) layer [40], rectified linear unit (ReLU) layer, and max-pooling layer with a stride of 2. The max-pooling layer is followed by repeated residual blocks in ResNet-34 (as shown by the green blocks in Fig. 2). All layers in the pretrained ResNet-34 network are kept except for the average pooling layer and the fully connecting layer at the end of the original network. These two layers are replaced with the decoder of the network. The decoder consists of repeating upsampling blocks (magenta and purple blocks in Fig. 2) that double the spatial resolutions of the output activations while

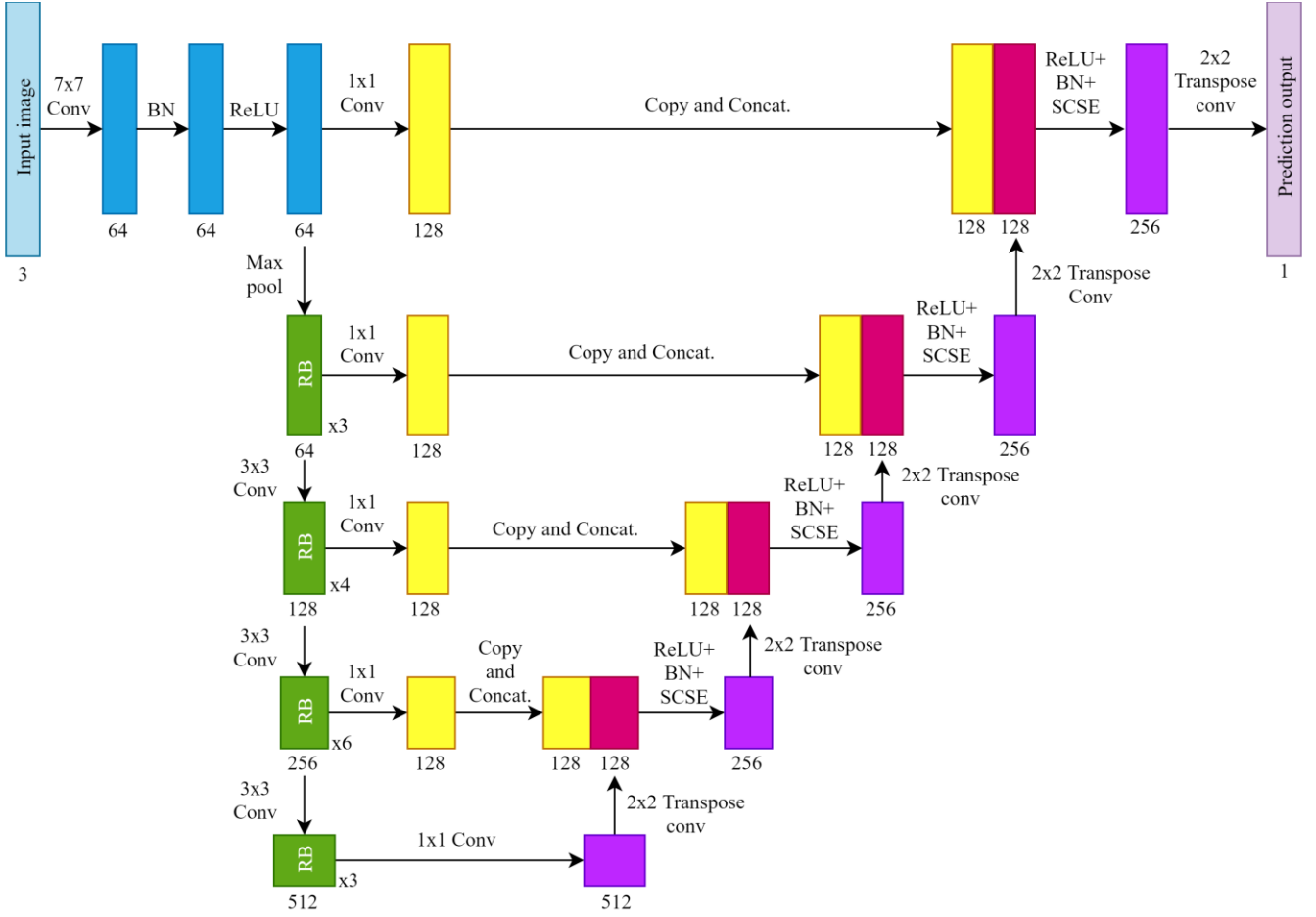


Fig. 2. Proposed network architecture. The blocks represent the activations whereas the arrows represent the layer operations. The green blocks represent the residual blocks (RB) in ResNet-34. For simplicity of visualization, the magenta blocks represent the activation before passing through successive layers of ReLU, batch norm (BN), and SCSE. The number below each block represents the number of channels it has.

halving the number of feature channels. Each upsampling block consists of a batch normalization layer, ReLU layer, and a transpose convolution layer of kernel size  $2 \times 2$  and a stride of 2, which provides upsampling. A concurrent spatial and channel squeeze and excitation (SCSE) module [41] is added between the batch normalization layer and the transpose convolution layer.

There are additional convolution layers, connecting several downsampling blocks to their corresponding upsampling blocks with the same spatial resolution, to perform a  $1 \times 1$  convolution on each downsampling block. The input of each upsampling block is a channel-wise concatenated tensor of both the output of the previous block and the output activation from the downsampling block. These operations are illustrated by the horizontal arrows connecting the yellow blocks in Fig. 2.

Because of the fully convolutional nature of the network architecture, it can receive virtually any image size larger than  $32 \times 32$ . We exploited this property to train various image sizes:  $128 \times 128$ ,  $256 \times 256$ , and  $320 \times 320$ . This adds a regularization effect to the neural network and provides it with more training images.

### B. Loss Function

We chose the *dice coefficient loss* [42] as the loss function because it directly optimizes the dice score, which is equivalent

to the *F1 score* [43]. This loss function is also used for our segmentation task because there is a severe class imbalance. In the pavement crack images, the non-crack pixels outnumber crack pixels by a ratio of around 65:1. The output layer of the network is passed through a sigmoid function so that each element in the output layer would have a range of  $[0,1]$ , which reflects the probability that a crack is present in each pixel:

$$\hat{y} = \sigma(z) \quad (5)$$

where  $\sigma(\cdot)$  denotes the element-wise sigmoid function,  $\hat{y} \in \mathbb{R}^{H \times W \times 1}$  denotes the prediction matrix in which each element represents the probability of a pixel being a crack pixel, and  $z \in \mathbb{R}^{H \times W \times 1}$  is the feature map of the final output layer. The dice coefficient loss  $L$  is determined by the following equation:

$$L = \frac{1}{N} \sum_{n=1}^N 1 - \frac{2|\hat{y} \odot y|}{|\hat{y}| + |y|} \quad (6)$$

where the operator  $|\cdot|$  denotes the sum of all matrix elements of its argument and  $\odot$  denotes the element-wise multiplication operation. The loss is a scalar value which is designed to output a value of 0 if the prediction completely matches the ground truth, and 1 if otherwise.

### C. Training Procedure

The specification of the workstation used to train the CNN is as follows: GTX 960M GPU (4 GB VRAM), Intel Core i5 processor, and 16 GB RAM. The deep learning framework used is PyTorch version 0.3.0 [44] and we also used the *fastai* library (version 0.6) [45] which was built on top of PyTorch. (We provide this information so that the reader can calibrate our results appropriately.)

#### 1) Parameter Initialization

The parameters in each convolutional layer in the upsampling part, which consists of weights and biases, were initialized using the method of He et al. [46]. The weights are sampled from a Gaussian distribution with a mean of 0 and a variance of  $2/n_l$ , where  $n_l$  is the number of elements in the kernel filter of the convolutional layer  $l$ , which also represents the number of connections of a response:  $n_l = k^2c$ , where  $k$  is the kernel filter size and  $c$  is the number of channels of the input activation. The bias was initialized to zero for all layers. As for the ResNet-34 encoder, we used the pretrained parameters.

#### 2) Parameter Optimization

After initialization, the parameters are then optimized to minimize the loss function using the AdamW [39] optimizer. A simplification of the AdamW optimizer is described as follows. First, at iteration  $t$ , the partial derivative of the loss function  $\mathbf{g}_t$  with respect to the parameters of the previous iteration  $\boldsymbol{\theta}_{t-1}$  is determined:

$$\mathbf{g}_t = \nabla_{\boldsymbol{\theta}} f_t(\boldsymbol{\theta}_{t-1}). \quad (7)$$

The first-moment estimate  $\mathbf{m}_t$  and second-moment estimate  $\mathbf{v}_t$  are computed as the convex combination of their respective estimates from the previous iteration and the partial derivative of the loss function:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t \quad (8)$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \mathbf{g}_t^2 \quad (9)$$

where  $\beta_1$  and  $\beta_2$  are hyperparameters which are set as 0.9 and 0.999 by default respectively. At  $t = 1$ ,  $\mathbf{m}_1$  and  $\mathbf{v}_1$  are calculated with the initial values of  $\mathbf{m}_0$  and  $\mathbf{v}_0$ , both 0.

Bias correction is done to reduce the effect of the term  $1 - \beta_1$  on  $\mathbf{g}_t$ , as well as to reduce the sensitivity to the initial values  $\mathbf{m}_0$  and  $\mathbf{v}_0$ , which are arbitrarily set to 0 before the first iteration. The bias correction is done according to

$$\hat{\mathbf{m}}_t = \mathbf{m}_t / (1 - \beta_1^t), \quad (10)$$

$$\hat{\mathbf{v}}_t = \mathbf{v}_t / (1 - \beta_2^t). \quad (11)$$

Finally, the parameters of the next iteration  $\boldsymbol{\theta}_t$  are updated using the following equation:

$$\boldsymbol{\theta}_t = (1 - \lambda) \boldsymbol{\theta}_{t-1} - \alpha \left( \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} \right) \quad (12)$$

where  $\lambda$  is the weight decay (set to 0.01),  $\alpha$  is the learning rate, and  $\epsilon$  is the epsilon number (set to  $10^{-8}$ ).

#### 3) Tuning the Learning Rate

The learning rate  $\alpha$  in the optimization procedure above is not fixed over the iterations or the same throughout the network. It changes across various layers in the network architecture and with each training iteration.

We divided the CNN into three groups of layers (*layer-groups*, for short). The first layer-group spans from the input to the 128-channel residual block. The second layer-group spans from the 256-channel residual block to the rest of the encoder. The third layer-group includes the entire upsampling part (the decoder). The three layer-groups received learning rates of ratio 1/9: 1/3: 1. The rationale for this ratio is that the earlier layers learn features that are more fundamental, such as oriented edges and corners, whereas the later layers learn more specific features, which are similar to real images [47]. Therefore, when the pretrained ResNet-34 was used for segmentation tasks, the earlier layers do not need to be trained as much as the latter layers.

Because we used transfer learning [48] in our network architecture, we fine-tuned the CNN in two stages. In the first stage, we froze the first layer-group, and its learning rate was temporarily set to zero so that all the parameters in it were not updated. The second and third layer-groups were trained normally for 15 epochs. In the second stage, we unfroze the first layer-group. Training was continued from the last training state from the first stage for an additional 30 epochs. In other words, the first layer-group was trained only after the other two layer-groups were well optimized. By training the latter parts of the pretrained ResNet-34 (second layer-group) and the decoder (third layer-group), this procedure takes maximum advantage of transfer learning. This is because these layers can be optimized for the segmentation task while keeping intact the earlier layers of ResNet-34, which are likely to have lower-level semantics.

Instead of using a constant learning rate throughout all epochs, we used a cyclical learning rate schedule that is similar to “one-cycle” training, proposed by [35, 36]. We determined the maximum learning rate,  $lr_{\max}$ , by using a systematic method, instead of the usual trial-and-error or grid-search methods used by other researchers. The first training batch was trained at a small learning rate, and then the following batches were trained with logarithmically increasing learning rates. We then plotted the graph of validation loss against the learning rate, as illustrated in Fig. 3. The maximum learning rate was chosen at the point before the validation loss increased sharply. An example is illustrated in Fig. 3, where the value of the maximum learning rate is chosen between  $10^{-3}$  to  $10^{-2}$ . Our cyclical learning rate regime is illustrated in Fig. 4 and proceeds as follows. The minimum learning rate value,  $lr_{\min}$ , is set to 5% of  $lr_{\max}$ . We then trained each batch with linearly increasing learning rates, which began with  $lr_{\min}$  and ended with  $lr_{\max}$  at about 40% of the total number of iterations. Beyond that point, we linearly decreased the learning rates. At about 85% of the total number of iterations, the learning rate returns to  $lr_{\min}$ . The learning rate then trickles down to near zero until the last iteration (see Fig. 4). The cyclical learning rate schedule was applied only once, regardless of the number of epochs. Note that the values for  $lr_{\min}$  and  $lr_{\max}$  above apply only to the third layer-group. The other two layer-groups were trained with



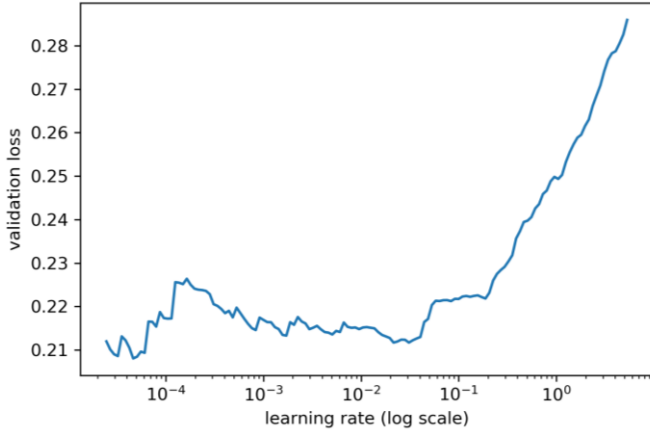


Fig. 3. A sample plot of the validation loss against learning rate. In this case, the maximum learning rate is chosen at any value between  $10^{-3}$  to  $10^{-2}$ .

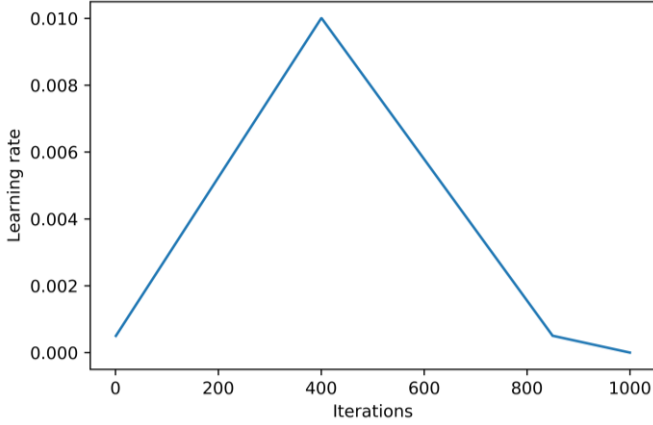


Fig. 4. An example of the cyclical learning rate schedule. In this example, the total number of iterations is 1000. The learning rate peaks at the 400<sup>th</sup> iteration. In practice, the actual number of iterations can be set to any value.

different (though similar) cyclical learning rate schedules, but their learning rates were scaled according to the aforementioned ratio of 1/9: 1/3: 1.

## V. RESULTS AND DISCUSSION

We evaluated the performance of our proposed neural network on the 46 images in the validation set, comparing our method with several other existing methods (see Table II for a list of the methods compared). A binary prediction matrix was obtained by setting all elements of  $\hat{y}$  (refer to (5)) with value of more than 0.5 to be 1 (crack), and 0 (non-crack) otherwise. Precision ( $Pr$ ), recall ( $Re$ ), and  $F1$  score were selected as the performance metrics. These metrics are commonly used in evaluating classification tasks. Their precise definitions are as follows:

$$Pr = \frac{TP}{TP + FP} \quad (13)$$

$$Re = \frac{TP}{TP + FN} \quad (14)$$

$$F1 = \frac{2 \times Pr \times Re}{Pr + Re} \quad (15)$$

where  $TP$ ,  $FP$ , and  $FN$  are the numbers of true positives, false positives, and false negatives, respectively. Note that the accuracy score was not used as an evaluation metric. This is because the true negatives outweigh the true positives significantly and are therefore easy to predict. Therefore, the number of true negatives does not properly reflect the quality of the prediction.

Because there are transition regions between the crack pixels and the non-crack pixels in the subjectively labeled ground truth, we consider 2 pixels in the vicinity of a labeled crack pixels as true positives. This evaluation convention was also used in other papers to evaluate their respective methods on the CFD dataset [7]. The  $Pr$ ,  $Re$ , and  $F1$  scores of all validation images were averaged. The final prediction matrix has one channel and a spatial dimension of  $320 \times 320$ . Hence, we cropped out nine  $320 \times 320$  images with a stride of 20 pixels for each image in the validation set, before evaluating them separately. The binary prediction masks were overlapped accordingly, and the prediction probabilities were averaged.

Table II shows a quantitative comparison of the precision, recall, and  $F1$  scores for our proposed method and the other methods tested on the CFD dataset [7]. (The best  $Pr$ ,  $Re$ , and  $F1$  scores are highlighted in bold.) Below is a summary of each compared method. [3] preprocessed images using morphological filters, before a dynamic thresholding was applied to filter out the darker pixels. In [49], a probability map for the crack prediction was constructed based on the pixel intensity and multi-scale neighborhood information. [50] extracted candidate cracks using a windowed intensity path-based method before segmenting the cracks using a crack evaluation model based on a multivariate statistical hypothesis test. [51] used a deep residual network as the encoder of the network, and a fully connected network as its decoder. [32], [33], and [34] also used U-Net-based architecture, similar to our proposed method, but with several modifications and different training schemes. [25] used a CNN-based classifier to predict the existence of crack in each  $5 \times 5$  image patch (as discussed in Section I).

From Table II, we can see that our proposed method achieves a significantly higher precision than the other compared methods. Although the recall is slightly lower than CrackForest [7] and the proposed method by Fan et al. [25], our proposed method achieves an overall  $F1$  score of 95.55%, higher than the other methods (see Table II).

Fig. 5 depicts a qualitative comparison of the methods. We can see that our neural network managed to capture most of the crack pixels, with some false negatives. Moreover, our method is not susceptible to noise in the image, such as oil stains and road paint. Even for images containing complex crack patterns, our neural network does a good job of predicting true positives. Unlike Canny [5], our network does not predict a lot of false positives, which is reflected in the high precision score. When compared to CrackForest, which overestimates the boundaries of the crack regions, our proposed method makes crack predictions that are very close to the boundaries in the ground

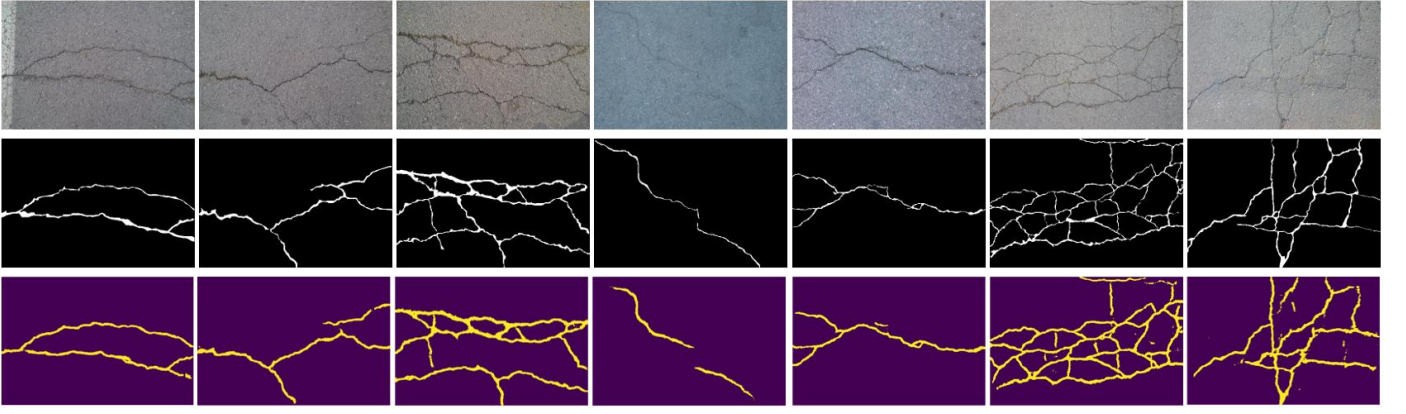


Fig. 5. Part of the prediction results from the validation set. From top to bottom row: original image, ground truth, binary prediction mask.

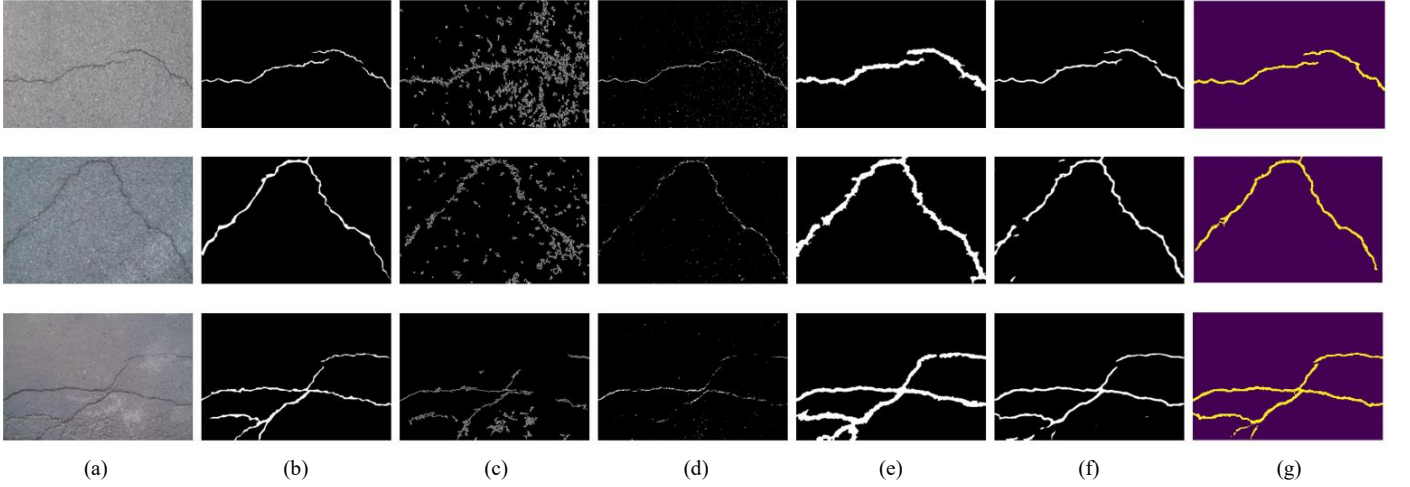


Fig. 6. Comparison of results among various methods. From left to right column: (a) original image, (b) ground truth, (c) Canny [5], (d) local thresholding [3], (e) CrackForest [7], (f) method of Fan et al. [25], (g) our method.

truth images. From Fig. 6, the neural network of Fan et al. [25] generally makes good predictions (at least visually), but makes wrong predictions for some of the pixels. For instance, in the second row of Fig. 6, the predictions made by Fan et al. [25] contain false positives at pixels far away from the ground truth, while also having false negatives at some crack regions.

TABLE II  
COMPARISON OF VARIOUS METHODS

Method	<i>Pr</i>	<i>Re</i>	<i>F1</i>
Canny [5]	0.4377	0.7307	0.4570
Local thresholding [3]	0.7727	0.8274	0.7418
CrackForest [7]	0.7466	0.9514	0.8318
Probabilistic generative model [49]	0.9070	0.8460	0.8700
U-Net by Jenkins et al. [32]	0.9246	0.8282	0.8738
MFC algorithm [50]	0.8990	0.8947	0.8804
FCN with residual network [51]	0.9357	0.8490	0.8903
U-Net by Nguyen et al. [33]	0.9306	0.8931	0.9114
Structural prediction with CNN [25]	0.9119	0.9481	0.9244
U-Net by Cheng et al. [34]	0.9212	<b>0.9570</b>	0.9388
Our method	<b>0.9702</b>	0.9432	<b>0.9555</b>

## VI. CONCLUSION

Because of the need to automate the segmentation tasks for pavement crack images, computer algorithms are often used. We have demonstrated that pavement crack segmentation tasks can be solved accurately using our deep learning technique. Our network architecture is a U-Net with an encoder of pretrained ResNet-34. A systematic method was used to find the optimum learning rate. In addition, we deployed the “one-cycle” training schedule to speed up the convergence. We also adopted techniques such as freezing layer-groups, assigning different learning rates to each layer-group, and increasing image sizes progressively. Our approach achieved an *F1* score of about 96% on the CFD dataset [7], outperforming the previous best method by a 1.7% margin. Our approach requires minimal feature engineering compared to other machine learning techniques; hence, it is particularly useful when there is a lack of domain expertise in analyzing pavement cracks. Our approach can be applied to various deep learning frameworks because it is not dependent on any special environments.

A limitation of this study is that our algorithm requires every pixel in the ground truth images to be manually labelled, which makes data acquisition expensive. One possible research direction to mitigate this issue is unsupervised learning-based

techniques, which they do not require any ground truth labels. Moreover, since supervised learning algorithms aim to fit the function approximator to the given labelled training data, the performance on real-life unseen data is largely dependent on how much the training dataset reflects the real-life pavement crack images. Hence, practical algorithm deployment potentially requires a much bigger training dataset, than the one we used, to capture data of a wider distribution. Further improvements can be done by collecting more high-resolution training images as well as high quality ground truth annotations. In addition to that, more studies can be done to discover more augmentation techniques to alleviate the problem of small datasets.

## REFERENCES

- [1] A. K. Gupta, and S. S. Adlinge, "Pavement deterioration and its causes," *IOSR-JMCE*, pp. 09-15, Jan. 2018.
- [2] E. Zaloshnja and T. R. Miller, "Cost of crashes related to road conditions, United States, 2006," (in Eng), *Ann. Adv. Automot. Med.*, vol. 53, pp. 141-153, Oct 2009.
- [3] H. Oliveira and P. L. Correia, "Automatic road crack segmentation using entropy and image dynamic thresholding," in *EUSIPCO*, Glasgow, Scotland, Aug. 2009, pp. 622-626.
- [4] N. Tanaka and K. Uematsu, "A crack detection method in road surface images using morphology," in *MVA*, Chiba, Japan, 1998, pp. 154-157.
- [5] H. Zhao, G. Qin, and X. Wang, "Improvement of Canny algorithm based on pavement edge detection," in *CISP*, Yantai, China, 2010, pp. 964-967.
- [6] V. Kaul, A. Yezzi, and Y. Tsai, "Detecting curves with unknown endpoints and arbitrary topology using minimal paths," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 10, pp. 1952-1965, Oct. 2012.
- [7] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen, "Automatic road crack detection using random structured forests," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 12, pp. 3434-3445, 2016.
- [8] Y. Zhang, "The design of glass crack detection system based on image preprocessing technology," in *ITAIC*, Chongqing, China, 2014.
- [9] M. Salman, S. Mathavan, K. Kamal, and M. Rahman, "Pavement crack detection using the Gabor filter," in *ITSC*, The Hague, Netherlands, 2013, pp. 2039-2044.
- [10] B. Shan, S. Zheng, and J. Ou, "A stereovision-based crack width detection approach for concrete surface assessment," *KSCE J. Civ. Eng.*, vol. 20, no. 2, pp. 803, 2016.
- [11] S. Iyer and S. Sinha, "A robust approach for automatic detection and segmentation of cracks in underground pipeline images," *Image Vis Comput.*, vol. 23, no. 10, pp. 921-933, 2005.
- [12] A. Talab, Z. Huang, F. Xi, and L. HaiMing, "Detection crack in image using Otsu method and multiple filtering in image processing techniques," *Optik*, vol. 127, no. 3, pp. 1030-1033, 2015.
- [13] Amhaz, R. *et al.*, "A new minimal path selection algorithm for automatic crack detection on pavement images," in *ICIP*, Paris, France, 2014, pp. 788-792.
- [14] T. Merazi-Meksen, B. Boudraa, R. Draï, and M. Boudraa, "Automatic crack detection and characterization during ultrasonic inspection," *J. Nondestruct. Eval.*, pp. 169-174, 2010.
- [15] Y. S. Yang, C. M. Yang, and C. W. Huang, "Thin crack observation in a reinforced concrete bridge pier test using image processing and analysis," *Adv. Eng. Softw.*, vol. 83, pp. 99-198, 2015.
- [16] Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang, "Crack Tree: automatic crack detection from pavement images," *Pattern Recognit. Lett.*, vol. 33, pp. 227-238, 2012.
- [17] H. Oliveira and P. Correia, "Automatic road crack detection and characterization," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, pp. 155-168, 2013.
- [18] R. G. Lins and S. N. Givigi, "Automatic crack detection and measurement based on image analysis," *IEEE Trans. Instrum. Meas.*, vol. 65, no. 3, pp. 583-590, 2016.
- [19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998,
- [20] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84-90, 2017.
- [21] E. Andre *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115, 2017.
- [22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, San Diego, CA, USA, May 2015.
- [23] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *NIPS*, Barcelona, Spain, Dec. 2016.
- [24] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640-651, 2017.
- [25] Z. Fan, Y. Wu, J. Lu, and W. Li, "Automatic pavement crack detection based on structured prediction with the convolutional neural network," *arXiv.org*, 2018.
- [26] A. Zhang *et al.*, "Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network," *Comput-Aided Civ. Inf.*, vol. 32, no. 10, pp. 805-819, 2017.
- [27] A. Zhang *et al.*, "Automated pixel-level pavement crack detection on 3D asphalt surfaces with a recurrent neural network," *Comput-Aided Civ. Inf.*, vol. 34, no. 3, pp. 213-229, 2019.
- [28] F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, and H. Ling, "Feature pyramid and hierarchical boosting network for pavement crack detection," *IEEE Trans. Intell. Transp. Syst.*, pp. 1-11, Apr. 2019.
- [29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, Las Vegas, USA, Jun. 2016, pp. 770-778.
- [30] C. Coleman *et al.*, "DAWNbench: an end-to-end deep learning benchmark and competition," in *NIPS ML Sys*, CA, USA, Dec. 2017.
- [31] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: convolutional networks for biomedical image segmentation," in *MICCAI*, Munich, Germany, Oct. 2017.
- [32] M. D. Jenkins *et al.*, "A deep convolutional neural network for semantic pixel-wise segmentation of road and pavement surface cracks," in *EUSIPCO*, Rome, Italy, Sept. 2018, pp. 2120-2124.
- [33] N. T. H. Nguyen, T. H. Le, S. Perry, and T. T. Nguyen, "Pavement crack detection using convolutional neural network," in *SoICT*, Danang, Vietnam, 2018, pp. 251-256.
- [34] Cheng *et al.*, "Pixel-level crack detection using U-Net," in *TENCON*, Jeju Island, Korea, Oct. 2018, pp. 462-466.
- [35] L. N. Smith, "Cyclical learning rates for training neural networks," in *WACV*, Santa Rosa, CA, USA, 2017, pp. 464-472.
- [36] L. N. Smith, "A disciplined approach to neural network hyperparameters: Part 1 -- Learning rate, batch size, momentum, and weight decay," *arXiv.org*, 2018.
- [37] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *ICLR*, San Diego, CA, USA, May 2015.
- [38] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," in *NIPS*, Long Beach, CA, USA, Dec. 2017.
- [39] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, New Orleans, Louisiana, USA, May 2019.
- [40] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *ICML*, Lille, France, Jul. 2015, pp. 448-456.
- [41] A. G. Roy, N. Navab, and C. Wachinger, "Concurrent spatial and channel squeeze & excitation in fully convolutional networks," in *MICCAI*, Granada, Spain, Sept. 2018.
- [42] F. Milletari, N. Navab, and A. Seyed-Ahmad, "V-Net: fully convolutional neural networks for volumetric medical image segmentation," in *3DV*, Stanford, CA, Oct. 2016, pp. 565-571.
- [43] Y. Sasaki, "The truth of the F-measure," *Teach Tutor Mater*, Oct. 2007.
- [44] A. Paszke *et al.*, "Automatic differentiation in PyTorch," in *NIPS*, Dec. 2017.
- [45] J. Howard *et al.*, "fastai," *GitHub repository*, 2018. Available: <https://github.com/fastai/fastai>.
- [46] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on ImageNet classification," in *IEEE ICCV*, Santiago, Chile, Dec. 2015.
- [47] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *ECCV*, Zurich, Switzerland, Sept. 2014, vol. 8689, pp. 818-833.



- [48] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *NIPS*, Montreal, Quebec, Canada, Dec. 2016.
- [49] D. Ai, G. Jiang, L. Kei, and C. W. Li, "Automatic pixel-level pavement crack detection using information of multi-scale neighborhoods," *IEEE Access*, vol. 6, pp. 24452-24463, 2018.
- [50] H. Li, D. Song, Y. Liu, and B. Li, "Automatic pavement crack detection by multi-scale image fusion," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, pp. 2025-2036, 2018.
- [51] S. P. Bang, H. Kim, Y. Yoon, and H. Kim, "A deep residual network with transfer learning for pixel-level road crack detection," *ISARC*, vol. 35, pp. 1-4, 2018.