IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 29, 2021, accepted September 5, 2021, date of publication September 9, 2021, date of current version September 17, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3111223

DAUNet: Deep Augmented Neural Network for Pavement Crack Segmentation

VLADIMIR POLOVNIKOV[1], DMITRIY ALEKSEEV[1], IVAN VINOGRADOV[1], AND GEORGE V. LASHKIA[2], (Member, IEEE)

[1]Faculty of Mechanics and Mathematics, Moscow State University, 119991 Moscow, Russia
[2]Faculty of Engineering, Chukyo University, Nagoya 470-0393, Japan

Corresponding authors: George V. Lashkia (lashkia@sist.chukyo-u.ac.jp) and Vladimir Polovnikov (polovnikov@intsys.msu.ru)

ABSTRACT Crack detection and measurement are essential tasks for maintaining and ensuring safety. Accurate crack detection is very challenging because of non-uniform intensity, poor continuity, and irregular patterns of cracks. The complexity of the background and variability in the data acquisition process also complicate the problem. Many approaches to crack detection have been proposed, but the accuracy of the detection leaves much to be desired. The aim of this study is to develop a practical crack detection method for real-time maintenance. We focus on a deep end-to-end and pixel-wise crack segmentation. We propose a lightweight U-Net-based network architecture with emphasis on the learning process. In order to verify the effectiveness of the proposed method, we conduct tests on publicly available pavement crack datasets and compare our model with state-of-the-art crack detection methods. Extensive experiments show that the proposed method effectively detects cracks in a complex environment, and achieves superior performance. The code and proposed model can be found in https://github.com/dvalex/daunet

INDEX TERMS Deep learning, crack detection, U-Net, pavement crack segmentation.

I. INTRODUCTION

Automatic crack detection is essential for effective maintenance systems. That is why it has attracted much attention for scientific research. Fan *et al.* [1] and Cao *et al.* [2] have conducted an extensive survey of the various crack detection algorithms. Among those in recent years, deep learning has been widely applied to crack detection, because of its excellent robust feature representation capability. Deep learning approaches can be divided roughly into patch-based classification and segmentation. In the patch-based approach the image is cropped to small patches, then a deep neural network is trained to classify each patch as crack or not. Patch-based algorithms are sensitive to the patch size and because these methods do not extract cracks at the pixel-level, in general, a post-processing step is required to separate crack pixels from the background. In contrast, in the segmentation approach, each pixel is classified as a crack pixel or a background pixel. These are the so-called end-to-end methods. The end-to-end methods have an encoder-decoder architecture. Encoder networks consisting of convolutional layers are applied to extracted features of cracks by forming feature maps. Decoder networks are employed to resize feature maps to the same size as the input image after a series of up-sampling and convolution layers. The output of a model is a crack prediction map, where crack regions have a higher probability and non-crack regions have a lower probability.

The segmentation approach has obvious advantages over the patch-based approach. It is more robust and maintains global context information. There is no need for post-processing and parameter tuning. It is an end-to-end method and crack measurements can be carried out directly on the output image.

At present, researchers have proposed a series of segmentation algorithms. Yang *et al.* [3] adopt Holistically-nested Edge Detection (HED) [4], an edge detection method as its backbone architecture. To enhance feature representation a feature pyramid module is introduced enriching lower-level layers' context information from higher-level feature maps. Similar to HED, the proposed network has the so-called side networks or parallel prediction maps, whose hierarchical boosting module makes the model focus on hard samples rather than easy samples. Selection of the proper objective or loss function is important for training crack detection

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaochun Cheng.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

networks. During the training, the loss function is computed over all pixels in a training image and crack map. However, the distribution of crack and non-crack pixels is heavily biased since most regions are non-crack pixels. Yang *et al.* [3] adopt the HED strategy and use sigmoid cross-entropy loss with class-balancing weights. Song *et al.* [5] propose a model with ResNet encoder. The encoder is followed by Multiscale Dilated Attention (MDA) module. In this module dilated convolution is used to get information from more distanced pixels. MDA allows collecting semantic information at multiple scales with relatively fewer parameters. In the decoder a feature fusion upsampling module is proposed, merging low-level and high-level features. For the loss function Song *et al.* [5] choose dice coefficient, which is a popular loss function for image segmentation tasks, since it expresses an overlap between two regions. Lau *et al.* [6] propose a U-Net-based network architecture. In the encoder, a pre-trained ResNet-34 neural network is adopted. The channel squeeze and excitation module is used in the decoder. Dice coefficient is used as the loss function. Training of various layers with different learning rates is conducted to improve convergence and performance. Fan *et al.* [1] propose a method consisting of U-net architecture with multi-dilation module and hierarchical feature learning module. The multi-dilation module located between the encoder and decoder uses the dilated convolutions described above. Hierarchical feature learning module inspired by [4] uses side networks and integrates multi-scale features from high to low-level convolutional layers. The loss function is similar to [3]. Liu *et al.* [7] use VGG 16 convolutional layers in the encoder. In decoder hierarchical features acquired from multiple layers are upsampled and used as a side output before being all fused at the end. The main difference with other models is that a post-processing step based on guided filtering and conditional random fields methods are applied to refine the network predictions. The loss function is similar to [3]. Wang *et al.* [8] use pre-trained DenseNet121 as an encoder. A feature pyramid attention module is inserted between the encoder and decoder, which combines low-level and high-level features. The decoder module upsamples by combining low-level and high-level features. The proposed architecture resembles U-net. The sum of cross-entropy loss and dice loss is selected as the loss function.

Reviewing the related literature, we notice several trends in crack segmentation models. Most of the proposed crack segmentation networks fuse features of different scales. This is a necessary step to improve crack detection performance since higher-level feature maps contain context information while lower-level feature maps include detailed information. Another trend is a U-net architecture [9]. This can be justified by the fact that U-net fuses encoder feature maps with the decoder and can generate accurate feature maps. Another trend is the dilated convolution module. Features in each region cannot be clearly classified as crack, without the surrounding information. One way to collect the surrounding information is to increase the size of the filters, which however leads to an increase in parameters and the computational cost. The dilated convolution helps to solve this problem: it can capture rich context information with fewer parameters involved. The most popular loss functions used in crack segmentation networks are cross-entropy and dice loss.

Although several methods proposed in the literature have shown satisfactory results there is still much room for improvements. The objective of this study is to develop a practical crack detector that can be used in real-time maintenance applications. A practical model has to have a lightweight architecture. Based on the reviewed literature we chose U-net architecture, and after extensive experiments, we selected Resnet18 as the encoder. Adding dilated convolution and attention modules didn't improve the performance. Therefore, we directed our efforts on the learning phase, and by proposing a new augmentation strategy and carefully choosing loss functions and learning rate strategy we developed a network that outperforms state-of-the-art crack detection methods. We call the proposed method DAUNet (Deep Augmented U-Net).

Proposed models have to be evaluated and compared. Several evaluation metrics are proposed in the literature. The standard metrics used are precision (Pr), recall (Re), and F-measure (F). Yang *et al.* [3] introduce ODS (Optimal Dataset Scale), OIS (Optimal Image Scale), which are boundary evaluation metrics from [10]. Since Pr and Re cannot appropriately demonstrate the overlapping extent between detected crack and ground truth, Yang *et al.* [3] introduce a new metric AIU (Average IoU (Intersection-over-Union)), which is the average IoU over all thresholds, and it illustrates the overall overlap extent between detections and ground truth.

The evaluation of metrics requires ground true data. Transition regions between crack pixels and non-crack pixels are difficult to annotate, and that is why the ground truth data has an annotation bias. Researchers have dealt with the bias problem differently. [5], [7], [11] consider manual labeling as ground truth, ignoring bias. In the cases where images are carefully labeled by experts, this approach can be considered reasonable. The majority of crack detection publications [1], [6], [8], [12]–[16] consider all predictions to be true positive pixels if a crack pixel is k pixels near to the manually labeled crack pixel. Depending on the research k vary from 2 to 5. However, this criterion is too loose and not precise enough to measure the detection results. If a pixel is predicted as background and it is near to the manually labeled crack pixel, the error occurs only when it coincides with the ground truth. Also, the topology of the predicted true positives may substantially differ from the ground truth, especially in wide crack cases. Yang *et al.* [3] adopt an evaluation procedure from the image boundary detection literature, where annotation bias is dealt with by computing the correspondence between detected and ground truth pixels. Martin *et al.* [10] present the algorithm for computing the correspondence between a thresholded machine boundary map and a human-labeled boundary map. The correspondence problem is

converted into a minimum cost bipartite assignment problem, where the weight between a machine boundary pixel and a human boundary pixel is proportional to their relative distance in the image plane. One can then declare all boundary pixels matched beyond some threshold d_{max} to be non-hits. We call d_{max} as maximum tolerance for the correct match. To solve the assignment problem Martin *et al.* [10] propose to use Goldberg's algorithm which is part of the CSA package and is the best-known algorithm for min-cost sparse assignments [17]. Since cracks are different from boundaries and solving an assignment problem is time consuming, in [3] during evaluation both crack detection and ground truth are thinned to one pixel wide before computing performance metrics.

From above it is clear that it is not appropriate to compare the performance based on the published results, and testing of models has to be conducted based on the same evaluation criteria and implementation. Therefore, for comparison, we choose the state-of-the-art deep learning segmentation model from [3]. There are reasons for this selection: the code and data are public and fair comparisons are possible, the performance of the proposed model is high and reliable.

In this paper, we propose a novel crack segmentation method based on encoder-decoder architecture to perform crack detection in an end-to-end way. The contributions of this paper are summarized below.

- Development of a lightweight model that impressively outperforms state-of-the-art heavyweight models on benchmark datasets.
- Introduction of an augmentation and loss function combination technique for crack images which helps to obtain marginally better performance.
- Proposal of a new measurement to evaluate crack detection methods. Since ODS and OIS measures are time consuming we introduce their simplified versions. These new measures allow quick evaluation of the performance and thus help shorten development time.

We use known techniques but find out new combinations that result in a practical algorithm, which makes a big difference in the real world. The contribution of the paper is the practical algorithm that provides overwhelming superior results compared to state-of-the-art methods.

II. PROPOSED METHOD

We formulate crack segmentation as a pixel-wise binary classification task. For a given input image I the proposed model generates a crack prediction map M, where crack regions have a higher probability and non-crack regions have a lower probability. In this section, we describe the implementation details of the proposed DAUNet.

A. NETWORK ARCHITECTURE

The network we propose has a U-Net based architecture [9]. U-Net is a convolutional neural network that was developed for biomedical image segmentation. The main challenges in biomedical image segmentation are: structures with low contrast, strong shape variations, weak borders, and a few annotated images. U-Net outperformed the best sliding-window convolutional networks on several challenges for segmentation of neuronal structures in electron microscopic stacks. Since we face similar challenges in crack segmentation we choose U-Net as a base of the proposed method.

U-Net consists of an encoder (downsampling path) and a decoder (upsampling path). The encoder is the first half of the architecture. It is usually a pre-trained classification network like VGG or ResNet, which is used to encode the input image into feature representations at multiple different levels. We use ResNet18 [18] as the encoder. We also tried VGG16, ResNet34, and ResNet50. Taking into account execution speed and accuracy, ResNet18 is a clear winner. The ResNet18 is pre-trained on ImageNet, and its last two classification layers are removed. The decoder is the second half of the architecture. The goal is to semantically project the discriminative features (lower resolution) learned by the encoder onto the pixel space (higher resolution) to get a dense classification. The decoder comprises of repeating upsampling blocks that double the spatial resolutions of the output activations while halving the number of feature channels. While upsampling, the decoder also concatenates the higher resolution feature maps from the encoder network with the upsampled features in order to better learn representations. Since upsampling is a sparse operation we need more detailed information from earlier stages to better represent the localization. Finally, the model restores the image resolution through an upsampling operation. This fully convolutional network receives an RGB image as input and produces a one-channel crack prediction map of the same size.

The generalization ability of deep learning methods depends on a large amount of training data, which is difficult to obtain. Data augmentation is regarded as an effective strategy to address this problem. Popular crack image augmentation methods include random rotations, flips, and changes in lighting [6]–[8]. In [3] the proposed FPHBN crack detector is trained on the training set of the CRACK500 dataset and then is tested on 5 bench-mark datasets, including the CRACK500. In this research, since we choose the FPHBN as a primary target for comparisons, we follow the same procedures as in [3]. We construct a crack detector with training images of the CRACK500 dataset using a better data augmentation method.

B. DATA PREPROCESSING AND AUGMENTATION

The CRACK500 dataset contains 500 full images of size around 2000 × 1500 pixels, split into 250 images of training data, 50 images of validation data, and 200 images of test data. Each crack image has a pixel-level annotated binary map. Due to the limited number of images, Yang *et al.* [3] initially crop each image into 16 non-overlapped image regions and the regions containing less than 1,000 pixels of crack are rejected. The final dataset is comprised of 1,896 crops for training, 348 for validation, and 1124 for testing. Once the

FPHBN [3] is trained, it is tested on the test data and other datasets for generalizability evaluation.

We use the same 250 full images for training, the same test data (1124 cropped images), and the same number of crops (1896 per epoch) in the training, but the cropping procedure differs as explained below. Original crops from the dataset can be successfully used to train a neural network as it is done in [3]. But we prefer to create our own crops from full images to make the training procedure more flexible and effective. Also, since a manually annotated binary map is not very accurate, we apply a soft labeling approach to make errors at edges less valuable.

As crack images are invariant to shifts, rotation, stretch and other geometrical transformations, we apply them all with randomly uniform selected parameters:

- Horizontal and vertical flips with the probability of 0.5
- Rotation according to a central point from $-90°$ to $90°$
- Translations up to 30% of height and width
- Scaling by coefficient in range of [0.5, 1, 5]

To avoid new edges appearing after transformations we use a ''mirror'' filling method. Also to improve tolerance to camera parameters and shooting conditions, we add some color augmentation:

- Modify contrast up to 5%
- Modify brightness up to 5%
- Change hue no more than $5°$ (hue is measured from $0°$ to $360°$)
- Change saturation up to 5%

We also tried various noise/color jittering augmentation and brightness gradient to simulate different ISO and lighting conditions. We excluded it from the final solution as the performance results became slightly lower. We propose the following data selection algorithm:

1) Select a pair (I, M) of full-size image and corresponding annotated binary map from the training dataset
2) Smooth M by 5×5 Gaussian blur filter
3) Select a random 512×512 size crop (I_1, M_1) from (I, M). Calculate the sum of all values in M_1. If this number is less than S_{min}, then the chosen crop is rejected (few crack pixels, too much background) and we repeat the step. To avoid an infinite loop, we limit the number of attempts to 1000.
4) Apply color and geometrical augmentations to (I_1, M_1)
5) Select random 256×256 size subcrop (I_2, M_2) from (I_1, M_1). This time no additional limits on crack pixels.
6) Stack (I_2, M_2) into batches and train the network

Our crops are random compared with [3] and are obtained online while training. It gives some advantages compared to the static crops used in [3] since the data varies. Subcroping the crop that passed geometrical augmentations is another feature of the proposed augmentation method that had a positive effect on training. Geometrical augmentations leave images with empty spaces that must be filled. By first choosing a bigger crop and then selecting a subcrop we can obtain an image that has less filled empty spaces. We also note that after precomputing the integral of an image, step 3 can be calculated in constant time.

C. LOSS FUNCTION AND PARAMETER OPTIMIZATION

We propose a two-stage training. In the first stage, we use Focal Loss with Adam optimizer and cosine annealing learning rate scheduler. Focal Loss [19] corrects a class imbalance in classification tasks. For example, in the CRACK500 dataset 99% is the background and only about 1% of pixels in ground truth are classified as cracks. Focal Loss adds a modulating term to cross-entropy loss, focusing learning on the hard rare class examples. The scaling factor is reduced to zero as confidence in the dominating correct class approaches 1. It automatically downscales simple examples during training and focuses the model on complex examples. Focal Loss multiplies the standard cross-entropy by a factor $-gt \cdot \alpha \cdot (1-p)^{\gamma}$, where gt stands for ground truth, p for the class probability, and α, γ are tunable parameters.

We use Adam optimizer, an adaptive learning rate optimization algorithm introduced by Kingma and Lei Ba [20]. Adam quickly gained popularity, however, after a while researchers noticed that despite superior training time, it does not converge to an optimal solution for some tasks. Several countermeasures were proposed. One popular approach is presented in [21], where AdamW with weight decay is introduced. AdamW is used in [6] for optimization of the crack segmentation neural network. However, our experiments show that the higher the decay, the less the overlap area between the predicted and the ground truth cracks. Loshchilov and Hutter [21] also propose AdamWR, an extension of AdamW by cosine annealing with restarts. Cosine annealing is introduced in [22]. It quickly cools down the learning rate according to a cosine schedule. Since our experiments with AdamW don't give satisfactory results, we apply cosine annealing directly to Adam. We use cosine annealing and also apply a linear warmup at the start of the training process. At each iteration s we update the learning rate as follows:

$$LR(s) = \begin{cases} LR_{wu} + s \cdot \dfrac{LR_{base} - LR_{wu}}{S_{wu}}, & \text{if } s \le S_{wu} \\ \dfrac{1}{2} \cdot LR_{base} \cdot \left(1 + \cos\left(\pi \cdot \dfrac{s - S_{wu}}{S_{tot} - S_{wu}}\right)\right), & \text{otherwise} \end{cases}$$

S_{wu} denotes the number of warm-up steps $S_{wu} = E_{wu} \times S_e$, where E_{wu} is the number of warmup epochs and S_e is the number of steps per epoch. LR_{base} stands for learning rate upper range, and LR_{wu} for lower range. S_{tot} denotes the total number of steps $S_{tot} = E_{tot} \cdot S_e$, where E_{tot} is the total number of epochs.

Focal Loss turns the model's attention towards the rare class and solves the problem of balance between positive and negative examples, as well as the balance between easy and hard examples. However, it makes predictions soft, the average prediction values become small. Also prediction peak values for crack pixels vary. This makes the selection of the

appropriate threshold a difficult task. To make the model more confident in the predictions we propose a second stage. In the second stage, we fine-tune the network using Dice Loss [23] with the SGD optimizer, however here we use a constant learning rate. Dice coefficient which is equivalent to the F1 score is frequently used in crack segmentation models. It is essentially a measure of overlap between predicted and ground true samples. After applying Dice Loss, the network's confidence in prediction increases, and results become close to 0, 1.

III. BENCHMARK RESULTS

We compare the proposed DAUNet with the FPHBN model from [3]. We choose FPHBN because it is a recently published state-of-the-art model, and its code and data are publicly available. Since our model is also publicly available, fair and transparent comparisons are possible. We use the same datasets and same training settings as in [3].

A. DATASETS

CRACK500 is the dataset proposed in [3]. All models are trained on CRACK500 training data. Testing is performed on the CRACK500 test data and other datasets as described below. German Asphalt Pavement Distress (GAPs) dataset is presented in [24]. It includes a total of 1,969 gray valued images, with various classes of distress including cracks. Since the GAPs annotations are bounding boxes, they cannot be used in pixel-wise crack prediction tasks. Yang *et al.* [3] manually select 384 crack images from the GAPs dataset and conduct pixel-wise annotation, creating GAPs384, which we use for tests. Cracktree200 is the dataset presented in [25], which includes 206 pavement images and has a pixel-wise annotation. Another dataset that we consider is CFD proposed in [12], consisting of 118 images and manually labeled crack contours. The last dataset considered is AEL [3], which is a combination of 3 small datasets, containing 58 pixel-wise annotated crack images.

B. COMPARED METHODS

We choose FPHBN [3] as the primary target for comparison. It is a state-of-the-art deep learning segmentation model proposed recently. Other models include HED [4], RCF [26], and FCN [27]. These are models used in [3] for comparison. HED and RCF are state-of-the-art edge detection models. FCN is a set of semantic segmentation models, from which FCN-8s a high-performance model is chosen. FCN-8s combines semantic information from a deep, coarse layer with information from a shallow, fine layer to produce accurate and detailed segmentations.

C. EVALUATION CRITERIA

We employ OIS, ODS, and AIU, the metrics used in [3]. We also propose sOIS and sODS which are simplified versions of OIS, ODS, correspondingly. OIS and ODS are boundary evaluation metrics from [10]. OIS is the aggregate F-measure on the data set for the best threshold in each image. ODS is the best F-measure on the data set for a fixed threshold. Since Yang *et al.* [3] didn't present precise OIS and ODS definitions for the crack case, and it took us a while to figure out the implementation details, we give detailed explanations of the metrics below.

Let I_g be a ground truth annotation and I_s be a prediction score of an image I. Let $t \in [0, 1]$ be a threshold, and I_b^t be a binary image obtained by thresholding I_s using threshold t. Let $\bar{I}_b^t$ and $\bar{I}_g$ be results of thinning of I_b^t and I_g respectively to one pixel wide. We construct a minimum cost bipartite graph and find correspondence between $\bar{I}_b^t$ and $\bar{I}_g$ (see [10] for the details), where the weight between corresponding pixels is proportional to their relative distance in the image plane. Let N_{tp}^t be the number of matched pixel pairs with weight less or equal than $d_{max} * D$, where D represents the image size dimension and it is equal to the length of the image diagonal. N_{tp}^t represents the number of true positives for the threshold t. Let N_p^t be the number of all positives in $\bar{I}_b^t$, and N_g^t be the number of ground truth pixels in $\bar{I}_g$. The precision, recall, and F-measure computed for the threshold t are defined as follows: $Pr^t = N_{tp}^t/N_p^t$, $Re^t = N_{tp}^t/N_g^t$, $F^t = 2\frac{Pr^t \cdot Re^t}{Pr^t + Re^t}$ The optimal F-measure for an image is defined as $F_{opt} = \max_t F^t, t \in \{0.01, 0.02, \ldots, 0.99\}$. OIS is the average of optimal measures of all images in the dataset.

To define ODS we accumulate true positives, all positives and ground truths for the whole dataset. Let S_{tp}^t be the sum of N_{tp}^t, S_p^t be the sum of N_p^t and S_g^t be the sum of N_g^t over all the images. The aggregate precision, recall, and F-measure are computed as follows: $Pr^{*t} = S_{tp}^t/S_p^t$, $Re^{*t} = S_{tp}^t/S_g^t$, $F^{*t} = 2\frac{Pr^{*t} \cdot Re^{*t}}{Pr^{*t} + Re^{*t}}$. ODS is defined as ODS $= \max_t F^{*t}, t \in \{0.01, 0.02, \ldots, 0.99\}$.

The AIU is first introduced in [3] and is computed on the prediction and ground truth without a thinning operation. The AIU of an image is defined as $\frac{1}{N_t}\sum_t \frac{N_{pg}^t}{N_p^t + N_g^t - N_{pg}^t}$ where N_t denotes the total number of thresholds $t \in \{0.01, 0.02, \ldots, 0.99\}$; for a given threshold t, N_{pg}^t is the number of pixels of an intersected region between the predicted and ground truth crack areas; N_p^t and N_g^t denote the number of pixels of predicted and ground truth crack regions, respectively. The AIU of a dataset is the average of the AIU of all images in the dataset.

The proposed new metrics sOIS and sODS are simplified versions of OIS and ODS, correspondingly. As in AIU, we don't perform thinning and pixel matching and compute metrics directly on I_b^t and I_g images. For a given threshold t, we calculate N_{pg}^t and define precision, recall, and F-measure for an image as follows: $Pr^t = N_{pg}^t/N_p^t$, $Re^t = N_{pg}^t/N_g^t$, $F^t = 2\frac{Pr^t \cdot Re^t}{Pr^t + Re^t}$ The optimal F-measure for an image is $F_{opt} = \max_t F^t$ and sOIS is the average of optimal measures of all images in the dataset. The definition of sODS follows similar steps. We calculate S_{pg}^t, which is the sum of N_{pg}^t over

the dataset. Then we calculate the aggregate precision, recal, and F-measure, which leads us to sODS. $Pr^{*t} = S^t_{pg}/S^t_p$, $Re^{*t} = S^t_{pg}/S^t_g$, $F^{*t} = 2\frac{Pr^{*t} \cdot Re^{*t}}{Pr^{*t}+Re^{*t}}$, $\text{sODS} = \max_t F^{*t}$.

D. SCALING

Various datasets have various crack width ranges and image resolution. Therefore, we can't expect a good performance of the CRACK500-trained neural network on other datasets. Generally speaking, a network detects cracks with some width range in pixels. In practical settings, shots are made with fixed special resolution. Shooting conditions should be adjusted to the training data's shooting conditions, so the range in pixels has to match the range of interest. To overcome this problem, we propose a different scaling for each dataset.

We tried several approaches to find an optimal scaling factor for each dataset. We computed the average crack width for each dataset and determined the scale rate, but it didn't lead to improved performance. Scaling all datasets to have the same average crack width is another approach, but it only slightly improved performance. Negative results can be explained by the fact that data distribution in real datasets depends not only on the thickness of the cracks but also on the image resolution, shooting conditions, camera parameters, etc. For example, artifacts of compression algorithms may appear and sharpness also can be lost at high magnifications. However, the assumption that scale is important allowed us to find a suitable scaling coefficient for each dataset. Since we do not have a training set defined other than the CRACK500 dataset, we select one representative image of each dataset, and by varying the scale we determine appropriate coefficients according to model performance, as seen in Fig. 1. For the performance measure, we use $\text{maxIoU} = \max_t \frac{N^t_{pg}}{N^t_p+N^t_g-N^t_{pg}}$. We choose maxIoU because it has higher values and sharper peaks than AIU. The determined coefficients are shown in Table 1. For example, the scaling coefficient of the CrackTree200 dataset is 1.4. That means we scale an input image by 1.4 (i.e. height and width increased by 1.4), apply neural network and then scale back result to original resolution. Scaling is performed using Lanczos interpolation over 8×8 pixel neighborhood. We do not alter the CRACK500 dataset, because the model is trained on its training set and therefore its scaling factor is 1.

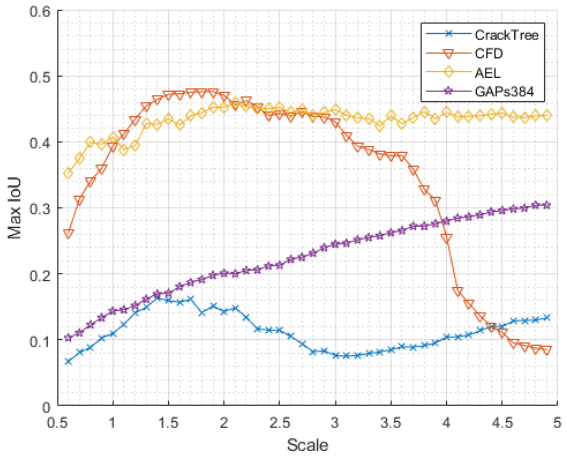


FIGURE 1. MaxIoU over scale curves of four datasets.

TABLE 1. Scale coefficient for each dataset.

Dataset	Scale
CRACKS500	1
CrackTree200	1.4
CFD	1.9
AEL	2.1
GAPS384	4.8

E. EXPERIMENTAL RESULTS

The proposed DAUNet is implemented on the widely used TensorFlow open-source library. The first stage of training is performed using 200 epochs, Focal Loss parameters are set to $\alpha = 0.2$, $\gamma = 0.25$. and learning rate parameters are set as follows: $LR_{base} = 0.001$, $LR_{wu} = 0$, $E_{tot} = 400$, $E_{wu} = 10$ and $S_e = 1896$. During the first 10 epochs, the learning rate increases after each batch linearly from 0 to 0.001. In the second stage, we fine-tune the network using Dice Loss using 100 epochs with a constant learning rate equal to 0.001. The augmentation parameter S_{min} is set to 3200000. Maximum tolerance allowed for correct matches of prediction and ground truth is set to the same value as in [3], $d_{max} = 0.0075$.

TABLE 2. Evaluation of Crack Detection Methods on the CRACK500 Test Dataset.

Methods	AIU	ODS	OIS	sODS	sOIS
DAUNet [proposed]	**0.565**	**0.676**	**0.706**	**0.750**	**0.731**
FPHBN [3]	0.489	0.604	0.635	0.647	0.591
HED [4]	0.481	0.575	0.625	N/A	N/A
RCF [26]	0.403	0.490	0.586	N/A	N/A
FCN [27]	0.379	0.513	0.577	N/A	N/A

Table 2 shows quantitative comparison results on the CRACK500 test dataset. The proposed DAUNet achieves the highest performance in all metrics. Relative to FPHBN, DAUNet improves performance by more than 12% in all metrics, and more than 14% in terms of AIU, sODS, and sIOS. In Fig. 2, we can see that DAUNet gains visually much precise crack detections than others.

From Table 3 we see that on the Cracktree200 dataset DAUNet improves performance by more than 39% in all metrics and outperforms FPHBN by more than 200% in AIU.

Table 4 shows results on the CFD dataset. Again DAUNet achieves better performance on all metrics. DAUNet improves more than 17% in all metrics and outperforms FPHBN by 113% in AIU.

As shown in Table 5, DAUNet achieves the best performance in terms of all metrics on the AEL dataset. From Fig. 2, we note that DAUNet achieves the most complete

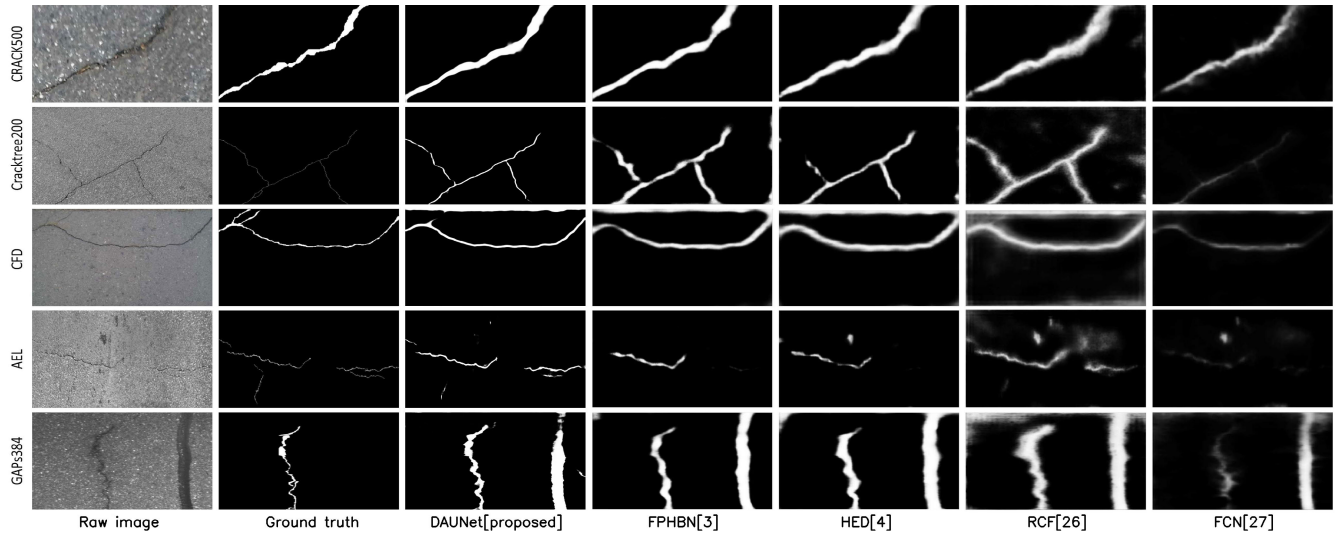


FIGURE 2. Examples of crack detection of compared methods on five datasets.

TABLE 3. Evaluation of Crack Detection Methods on the CrackTree200 Test Dataset.

Methods	AIU	ODS	OIS	sODS	sOIS
DAUNet [proposed]	**0.128**	**0.781**	**0.805**	**0.234**	**0.276**
FPHBN [3]	0.041	0.517	0.579	0.095	0.125
HED [4]	0.040	0.317	0.449	N/A	N/A
RCF [26]	0.032	0.255	0.487	N/A	N/A
FCN [27]	0.008	0.334	0.333	N/A	N/A

TABLE 4. Evaluation of Crack Detection Methods on the CFD Test Dataset.

Methods	AIU	ODS	OIS	sODS	sOIS
DAUNet [proposed]	**0.370**	**0.812**	**0.831**	**0.603**	**0.593**
FPHBN [3]	0.173	0.683	0.705	0.377	0.372
HED [4]	0.154	0.593	0.626	N/A	N/A
RCF [26]	0.105	0.542	0.607	N/A	N/A
FCN [27]	0.021	0.585	0.609	N/A	N/A

TABLE 5. Evaluation of Crack Detection Methods on the AEL Test Dataset.

Methods	AIU	ODS	OIS	sODS	sOIS
DAUNet [proposed]	**0.223**	**0.615**	**0.660**	**0.400**	**0.394**
FPHBN [3]	0.079	0.492	0.507	0.319	0.283
HED [4]	0.075	0.429	0.421	N/A	N/A
RCF [26]	0.069	0.469	0.397	N/A	N/A
FCN [27]	0.022	0.322	0.265	N/A	N/A

crack detection results. DAUNet improves FPHBN by more than 25% in terms of all metrics and increases AIU by 182%.

From Table 6 we see that the proposed DAUNet outperforms FPHBN on the GAPs384 dataset. DAUNet improves performance by more than 48% in all metrics. However, the performance is much lower than on the CRACK500 dataset. The GAPs384 dataset has a background that is similar to a crack. In the last row of Fig. 2, a sealed crack is misclassified as a crack. Even though the ground truth of GAPs384 is one or several pixels wide, the AIU value of DAUNet has a 167% increase.

TABLE 6. Evaluation of Crack Detection Methods on the GAPs384 Test Dataset.

Methods	AIU	ODS	OIS	sODS	sOIS
DAUNet [proposed]	**0.217**	**0.514**	**0.342**	**0.349**	**0.388**
FPHBN [3]	0.081	0.220	0.231	0.121	0.156
HED [4]	0.069	0.209	0.175	N/A	N/A
RCF [26]	0.043	0.172	0.120	N/A	N/A
FCN [27]	0.015	0.088	0.091	N/A	N/A

To further compare the proposed and state-of-the-art methods, we follow [3] and conduct experiments on images with a complex background, low illumination, and shadow. Fig. 3 shows results on the same images used in [3]. In low illumination condition (the second row of Fig. 3), all methods fail to detect a crack. This is because there are no similar images in the training data. On other images, we can see that the proposed DAUNet yields a clearer result and produces much fewer false positives.

Intersection over Union (IoU) curves of DAUNet and FPHBN are shown in Fig. 4. We see that the proposed DAUNet always has considerably higher IoU over various thresholds, showing higher accuracy in detecting cracks. We also apply image scaling approach to FPHBN. As we can see from Fig. 4, scaling significantly improves FPHBN performance on datasets where the model is not trained. These results confirm the effectiveness of the scaling approach. Despite significant improvements in FPHBN performance, DAUNet still retains its overall superiority.

The proposed DAUNet achieves the highest AIU, sODS and sOIS values on the CRACK500 dataset. This is because this dataset is the one selected for training the model. But at the same time, the cracks in them, as can be seen from Fig. 2, are wider than in other datasets. Therefore, the prior probability of getting a larger IoU by drawing a random crack of the corresponding width is higher in that case. The same applies to the sODS and sOIS metrics, which perform an exact pixel-by-pixel comparison. In contrast to sODS and

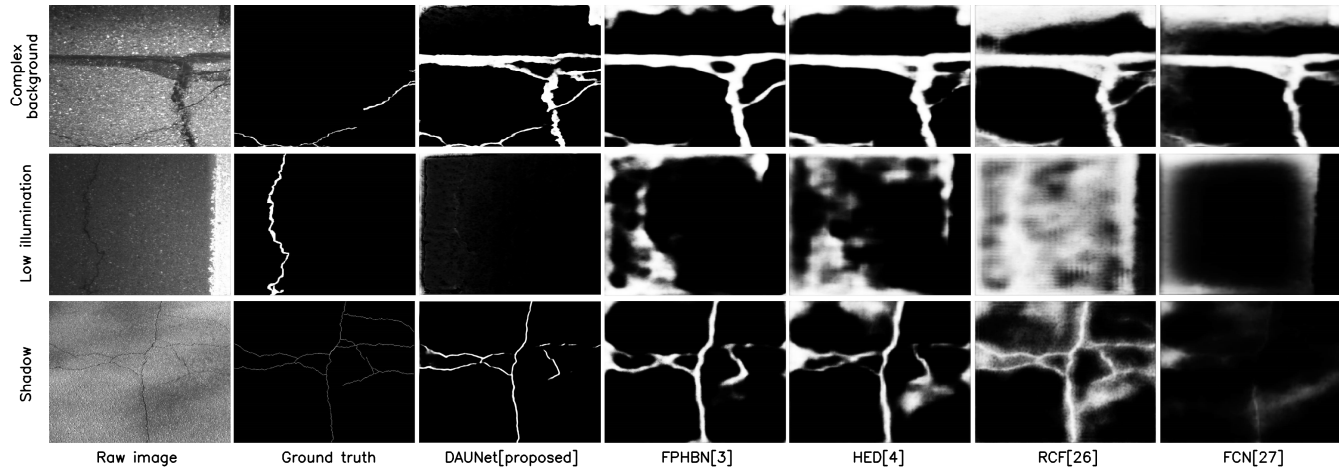


FIGURE 3. The visualization of detection results of compared methods on special cases, i.e., shadow, low illumination, and complex background.

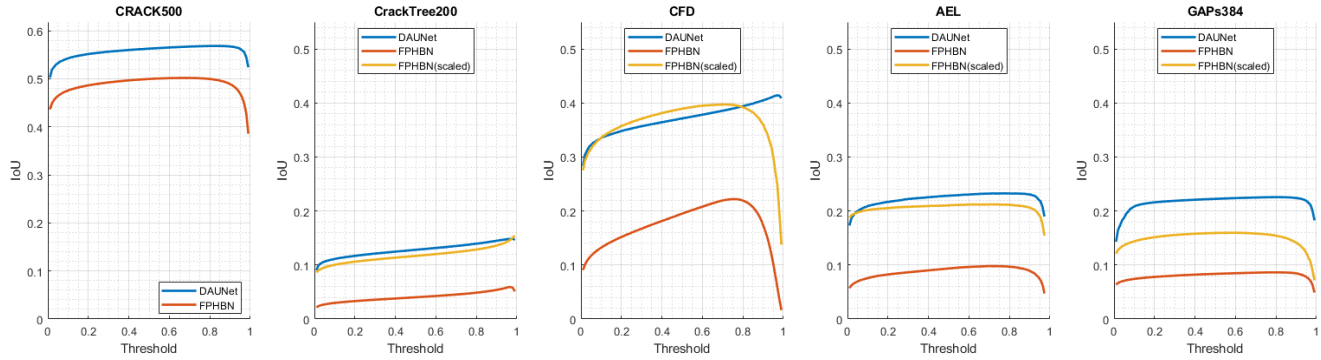


FIGURE 4. Intersection over Union of compared methods on five datasets.

sOIS, ODS and OIS metrics turn out to become relatively large on datasets with a small crack width. In our opinion, the reason is the mechanism of the tolerance of annotation inaccuracies. In other words, the predicted segmentation may not intersect with the real one at all, but lie somewhere nearby and have a different shape, but the matching will still be established.

The performance on the CRACK500 dataset could be even higher if the manually annotated maps were more accurate. Some annotations are shifted compared to real cracks, both in training and test subsets, and there are also some misinterpretations.

We also compare the inference execution speed of DAUNet and FPHBN on the CRACK500 dataset. The execution is conducted on GeForce GTX 1060 6GB. The average execution time of FPHBN is 596ms, while DAUNet achieves 66ms, and thus 9 times faster than FPHBN. The amount of parameters in FPHBN is 44.7 million, and DAUNet has just 14.3 million parameters.

IV. ABLATION STUDIES

We perform ablation studies of the proposed method on the CRACK500 dataset to show how some features affect the performance. Several experiments were carried out by varying loss functions and training procedures. We select loss functions that are popular in the crack segmentation task. These are: Focal Loss (FL), Binary Cross-Entropy (BCE), Dice Loss (DL), and Jaccard Loss (JL). We denote FL+DL two-stage optimization proposed in this paper. We replace FL by BCE, and consider BCE+DL. We also consider one-stage optimization using only DL or JL. Each of the cases is used in 5 different training procedures. We denote "Fine" a training process that uses the proposed augmentation and data selection algorithm. "FPHBN" denotes a training process that uses similar data preparation as in [3], however the size of the crops are chosen closest to the size used in [3] that satisfies requirements of U-Net. "FPHBN2" is the same as "FPHBN" however crops are resized to 256×256. "NoInit" doesn't use pre-trained weights, and "LR/10" denotes a similar strategy used in [3] that divides learning rate by 10 after each 25% of iterations. Note that in each training process non-mentioned parameters remain the same as in the proposed method. The results are shown in Table 7, 8, and 9.

The results indicate that the main factor that affects the performance is the proposed augmentation procedure. The selection of the loss function has a much less effect on the performance, but the best performance is achieved by two-stage optimization.

TABLE 7. AIU performance comparison between different training strategies.

AIU	FL+DL	BCE+DL	DL	JL
Fine	**0.565**	0.562	0.561	0.563
FPHBN	0.505	0.504	0.504	0.500
FPHBN2	0.501	0.498	0.507	0.508
NoInit	0.548	0.551	0.547	0.547
LR/10	0.554	0.552	0.553	0.553

TABLE 8. ODS/OIS performance comparison between different training strategies.

ODS/OIS	FL+DL	BCE+DL	DL	JL
Fine	**0.676** / 0.706	**0.676/0.710**	0.669/0.680	0.672/0.683
FPHBN	0.644/0.668	0.642/0.663	0.641/0.650	0.637/0.645
FPHBN2	0.629/0.651	0.626/0.652	0.636/0.651	0.637/0.651
NoInit	0.658/0.693	0.659/0.693	0.654/0.665	0.653/0.664
LR/10	0.665/0.701	0.660/0.693	0.659/0.671	0.657/0.672

TABLE 9. sODS/sOIS performance comparison between different training strategies.

sODS/sOIS	FL+DL	BCE+DL	DL	JL
Fine	**0.750** / 0.731	0.749/ **0.732**	0.741/0.708	0.743/0.709
FPHBN	0.714/0.675	0.711/0.672	0.702/0.652	0.700/0.648
FPHBN2	0.708/0.681	0.704/0.677	0.708/0.668	0.708/0.669
NoInit	0.735/0.724	0.737/0.724	0.732/0.696	0.732/0.694
LR/10	0.741/0.730	0.738/0.724	0.735/0.702	0.735/0.704

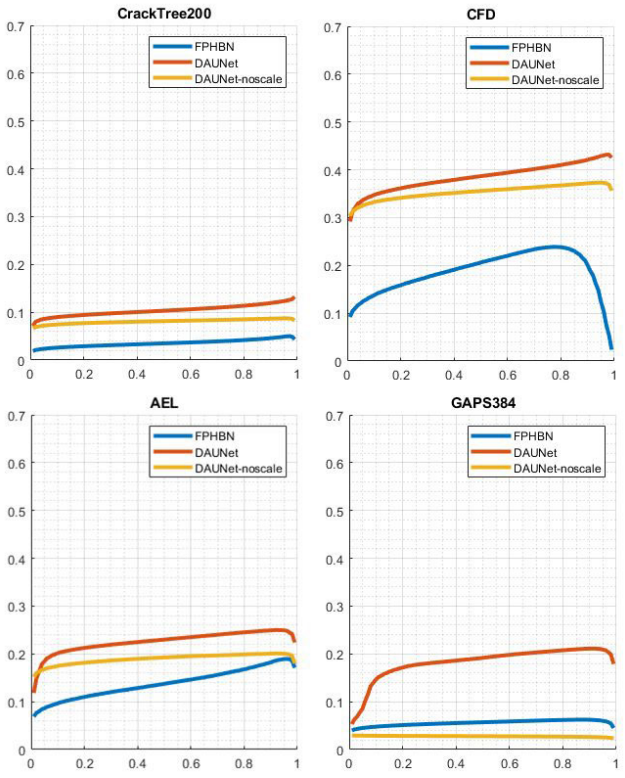


FIGURE 5. Scaling effect.

Next, we show how scaling affects performance. We already see in Fig. 4, that scaling significantly improves FPHBN performance on datasets where the model is not trained. As we can see from IoU curves of Fig. 5, removing scaling from consideration worsens DAUNet performance.

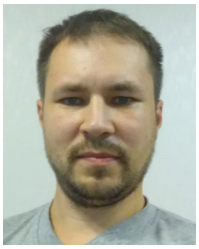
V. CONCLUSION

In this paper, we put our efforts into the learning phase, and by proposing a new augmentation strategy and carefully choosing loss function and learning rate scheduling we develop a network (DAUNet) that impressively outperforms the FPHBN crack detection method in terms of performance and execution speed. We choose FPHBN [3] as the primary target for comparison since it is a recently published state-of-the-art crack detection method and its code and data are public. The proposed DAUNet is also publicly available. Therefore, all results are transparent and can be easily confirmed.

REFERENCES

[1] Z. Fan, C. Li, Y. Chen, J. Wei, G. Loprencipe, X. Chen, and P. Di Mascio, "Automatic crack detection on road pavements using encoder-decoder architecture," *Materials*, vol. 13, no. 13, p. 2960, Jul. 2020.
[2] W. Cao, Q. Liu, and Z. He, "Review of pavement defect detection methods," *IEEE Access*, vol. 8, pp. 14531–14544, 2020.
[3] 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.*, vol. 21, no. 4, pp. 1525–1535, Apr. 2020.
[4] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1395–1403.
[5] W. Song, G. Jia, D. Jia, and H. Zhu, "Automatic pavement crack detection and classification using multiscale feature attention network," *IEEE Access*, vol. 7, pp. 171001–171012, 2019.
[6] S. L. H. Lau, E. K. P. Chong, X. Yang, and X. Wang, "Automated pavement crack segmentation using U-Net-based convolutional neural network," *IEEE Access*, vol. 8, pp. 114892–114899, 2020.
[7] Y. Liu, J. Yao, X. Lu, R. Xie, and L. Li, "Deepcrack: A deep hierarchical feature learning architecture for crack segmentation," *Neurocomputing*, vol. 338, pp. 139–153, Apr. 2019.
[8] W. Wang and C. Su, "Convolutional neural network-based pavement crack segmentation using pyramid attention network," *IEEE Access*, vol. 8, pp. 206548–206558, 2020.
[9] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. MICCAI*, Munich, Germany, 2017, pp. 234–241.
[10] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 530–549, May 2004.
[11] W. Choi and Y.-J. Cha, "SDDNet: Real-time crack segmentation," *IEEE Trans. Ind. Electron.*, vol. 67, no. 9, pp. 8016–8025, Sep. 2020.
[12] 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, Dec. 2016.
[13] R. Amhaz, S. Chambon, J. Idier, and V. Baltazart, "Automatic crack detection on two-dimensional pavement images: An algorithm based on minimal path selection," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2718–2729, Oct. 2016.
[14] Z. Fan, Y. Wu, J. Lu, and W. Li, "Automatic pavement crack detection based on structured prediction with the convolutional neural network," 2018, *arXiv:1802.02208*. [Online]. Available: http://arxiv.org/abs/1802.02208
[15] D. Ai, G. Jiang, L. S. Kei, and C. Li, "Automatic pixel-level pavement crack detection using information of multi-scale neighborhoods," *IEEE Access*, vol. 6, pp. 24452–24463, 2018.
[16] 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, no. 6, pp. 2025–2036, Jun. 2019.
[17] A. V. Goldberg and R. Kennedy, "An efficient cost scaling algorithm for the assignment problem," *SIAM J. Discrete Math.*, vol. 71, no. 2, pp. 153–177, Dec. 1995.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[19] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020.
[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[21] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. ICLR*, 2019, pp. 1–19. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7
[22] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," 2016, *arXiv:1608.03983*. [Online]. Available: http://arxiv.org/abs/1608.03983
[23] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation," 2016, *arXiv:1606.04797*. [Online]. Available: http://arxiv.org/abs/1606.04797
[24] M. Eisenbach, R. Stricker, D. Seichter, K. Amende, K. Debes, M. Sesselmann, D. Ebersbach, U. Stoeckert, and H.-M. Gross, "How to get pavement distress detection ready for deep learning? A systematic approach," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 2039–2047.
[25] Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang, "CrackTree: Automatic crack detection from pavement images," *Pattern Recognit. Lett.*, vol. 33, no. 3, pp. 227–238, 2012.
[26] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5872–5881.
[27] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.



VLADIMIR POLOVNIKOV received the Ph.D. degree in mathematics from Moscow State University, Russia, in 2008. He is currently a Research Scientist with the Faculty of Mechanics and Mathematics, Moscow State University. His research interests include artificial neural networks, especially mathematical representations of neural functions, machine learning, data mining, computer vision, pattern recognition, and algorithms.

DMITRIY ALEKSEEV received the Ph.D. degree in mathematics from Moscow State University, Russia, in 2006. He is currently a Senior Research Scientist with the Faculty of Mechanics and Mathematics, Moscow State University. His current research interests include pattern recognition, computer vision, and machine learning.

IVAN VINOGRADOV is currently pursuing the Ph.D. degree with the Faculty of Mechanics and Mathematics, Moscow State University. He is also a Research Staff Member with the Laboratory of Innovation of Medical Technologies. His current research interests include automata theory, computer vision, and machine learning.

GEORGE V. LASHKIA (Member, IEEE) received the Ph.D. degree from Moscow State University. He is currently a Professor with the School of Engineering, Chukyo University, Japan. His current research interests include computer vision, machine learning, and algorithms.