Effective Pavement Crack Delineation Using a Cascaded Dilation Module and Fully Convolutional Networks

Yasmina Benkhoui1, Tahar El-Korchi2, and Ludwig Reinhold1

1 Department of Electrical and Computer Engineering, WPI, Worcester, USA
ybenkhoui@wpi.edu
2 Department of Civil and Environmental Engineering, WPI, Worcester, USA

Abstract. Crack detection in concrete surfaces is a critical structural health monitoring task. In fact, cracks are an early indication of the decaying of the structure that can lead to severe consequences. Manual inspection is time-consuming, costly, and contingent on the subjective judgment of inspectors. To address these challenges, we propose to use state-of-the-art techniques in computer vision to approach the crack delineation problem as a semantic segmentation task where pixels of the same class (background or crack) are clustered together. Our proposed method uses dilated convolution to enlarge the receptive field and preserve the spatial resolution. In this work, we present a fully convolutional network that consists of an encoder, a cascaded dilation module, and a decoder. While the encoder extracts the feature maps from input images, the cascaded dilation module aggregates multi-scale contextual information and finally, the decoder fuses low-level features, performs pixel-wise classification, restores the initial resolution of the images and subsequently outputs the segmentation results. Based on the same meta-architecture, we compare three different dilated encoder-decoder (DED) models: DED-VGG16, DED-ResNet18, and DED-InceptionV3. The three models have been trained and validated using a dataset comprised of 40000 images. For evaluation purposes, we used common performance metrics for semantic segmentation tasks: Precision, Recall, F1-score, IoU, and ROC curves. Our results show that DED-VGG16 achieved the highest accuracy (91.78%) and generated precise visual semantic segmentation results.

Keywords: Semantic segmentation · Deep convolutional neural networks · Dilated convolution · Fully convolutional networks · Crack detection · Critical infrastructure

1 Introduction

According to the U.S. Department of Transportation 2018 National Bridge Inventory (NBI) database [1], 47,052 bridges are classified as structurally deficient or functionally obsolete. More often than not, concrete is used for bridge
superstructure components such as decks, curbs, sidewalks and pre-stressed concrete beams. Consequently, crack identification in concrete is a crucial aspect of structural health monitoring and serviceability evaluation. However, bridge inspection in the United States is not only complicated but also costly. In fact, this process relies primarily on visual inspection which cannot guarantee accuracy nor reliability and might result in inadequate structural integrity assessment since it is error-prone and dependent on the human judgement. Early research efforts [2] on detecting cracks are mainly built upon image processing techniques (IPTs) such as edge detection (Sobel, Canny, fast Fourier transform, fast Haar transform etc.), image filtering [3] and histogram analysis [4]. While these methods are able to identify cracks, their orientation and width, they are still significantly affected by lighting conditions and image noise; they tend to perform poorly in real-life inspection scenarios. To overcome the aforementioned drawbacks, researchers used IPTs as feature extractors and combined them with machine learning algorithms to identify specific crack characteristics such as depth, width and location within the image. Examples include but are not limited to regionally enhanced multi-phase segmentation technique for damaged surfaces [5], spatially tuned robust multi-feature [6], textural pattern recognition [7] and local directional pattern features [8]. In [9], support-vector machine (SVM) was utilized to identify images with or without cracks from concrete by extracting hand engineered features. The issue in these scenarios is that cracks are idealized and simplified which does not lead to robust crack detection. Other techniques include using unsupervised learning such as k-nearest neighbor [10] where extracted crack pixels are clustered. The advantage of unsupervised learning is that the model does not call for manually annotated data, however, performance is compromised especially under adverse lighting and shading conditions. To address these limitations, Computer Vision and specifically Convolutional Neural Networks (referred to as CNN or ConvNet) are widely used in image classification [11], semantic segmentation [12] and object recognition [13]. The recent surge of interest in deep learning methods is due to its proven ability to outperform previous state-of-the-art techniques [14,15]. Deep convolutional neural networks [16] automatically learn a set of network weights to extract features that are needed to achieve a certain task. In fact, in recent years, state-of-the-art DCNN have outperformed traditional methods and humans in tasks such as edge detection [17] where DCNN successfully labeled crack images at a rate of 99% compared to edge detection techniques which accurately detected 79% of cracked pixels. This outperformance is due to the fact that in DCNNs [18], the features automatically learned during the supervised learning process are more representative of the textural characteristics of the images compared to the features extracted using traditional IPTs.

Semantic segmentation consists of labelling each pixel in an image which leads to a better localization of cracks without using bounding boxes. It can be regarded as classification at the pixel level. Long et al. [19] utilized the classification capabilities of the modern DCNN architectures (AlexNet, VGG-16, GoogLeNet, ResNet) and modified them to get the output granularity required
for such a task. This was done by converting the fully connected layers into fully convolutional networks; it consisted of casting the fully connected layers to convolutions with kernels that are equal to their fixed input regions.

Efforts have recently focused on crack detection based on pixel-wise classification. Fully Convolutional Networks (FCNs) [20] and Auto-Encoders [21] are the most popular methods to solve this problem. In [20], the authors proved that the combination of dilated convolution and FCN improves the accuracy of semantic segmentation. The network used ResNet18 and ResNet50 as a backbone and compared their results. The generated feature maps output to five branches where different dilation rates are applied and then concatenated. In our work, we present a cascaded dilation module that works sequentially on the feature maps generated instead of following a parallel process. We show further in this paper the logic behind this reasoning and how it improves the fine details of the crack pixels. In [21], a two-stage algorithm is proposed where the first stage uses two-cascaded auto-encoders to segment the defects while the second stage focuses on classifying the cropped defect regions using a CNN classifier. Furthermore, in [22], the authors present an Encoder-Decoder based method, DeepCrack which is a deep hierarchical network that performs semantic segmentation to detect pavement crack. This architecture is based on SegNet [23] encoder-decoder model. The features generated are fused in pairs at the same scale and concatenated to generate a final fused output. While DeepCrack performs well in extracting thinner cracks, some width information is lost. Similarly, SegNet uses classification networks directly for pixel-wise segmentation. This results in a low resolution output mainly due to max-pooling and subsampling which reduces the feature map quality.

Our work presented in this paper falls into this same context: our aim is to accurately delineate the crack in concrete while preserving high resolution information. To serve this purpose, we propose a dilated convolutional FCN encoder-decoder network based architecture. The contributions of this paper can be summarized as follows:

- We developed an end-to-end trainable crack segmentation network. The designed model is an FCN encoder-decoder equipped with a dilated module that controls the receptive field. The changes of dilation rates effectively enlarge the kernel size without extra computation. As a result, the proposed network is able to obtain more abundant and high spatial resolution features.
- We present three different models based on the same meta-architecture by adopting three different encoder backbones: ResNet, VGG16 and InceptionV3.
- We improve the loss function to offset the highly unbalanced class problem related to the presence of a larger portion of positive pixels compared to negative ones. Positive pixels refer to crack pixels detected whereas negative pixels refer to non-crack pixels.
- We evaluate the crack delineation proposed framework by leading a complete study based on relevant metrics such as Precision, Recall, F-score, ROC curves and IoU.
Fig. 1. An illustration of the different modules of the proposed encoder-decoder meta-architecture. This FCN overview shows the three backbone encoders explored in this work (VGG16, ResNet18 and InceptionV3), the cascaded dilation module and the Decoder. We modified the encoder backbone so that we can execute dense prediction tasks by omitting the down-sampling operation in the last two layers. The feature maps generated are then forwarded to the cascaded dilation module which captures information at different spatial scales. Finally, the low-level features obtained by the skip connections are fused with the captured information and the decoder recovers the initial dimensions of the input images.

The remainder of this paper is structured as follows: Sect. 2 provides an overview of the models and methods used for the design of the system. Section 3 describes the implementation details, evaluation metrics used and discusses the results obtained. Finally, we conclude our findings in Sect. 4.

2 Models and Methods

2.1 Overview

In this section, we present the architecture of the FCN proposed for the task of crack delineation. The processing pipeline is illustrated in Figure 1. The encoders used are modified versions of VGG16, ResNet18, and InceptionV3 where the final layers are replaced by the cascaded dilation module. This is needed since the original DCNNs used as encoders will produce a low dimensionality dense representation of the input image; however, they lack fine details which are crucial in our case for crack detection. As illustrated in Sect. 3, there is a high imbalance between the background and the crack pixels in the dataset samples, which results in a biased classifier to the background class during the training process and consequently, generates poor segmentation results. To resolve this problem, we take into account the frequency of each class and use it to improve the softmax cross entropy loss function. Let us assume that the frequency of a given class \( l \) in the training data is \( f_l \) and that the sum of frequency of all classes (background and crack) is 1, i.e.: \( \sum_l f_l = 1 \). We add the inverse frequency of
each class to the cross entropy loss function which accounts for the pixels with less frequency classes. The cross entropy loss function is given by:

$$\text{loss}_{ce} = -\frac{1}{N} \sum_{i=1}^{n} \frac{1}{f_{y_i}} y_i \log(p_i) \quad (1)$$

where $N$ is the total number of pixels in an image, $y_i$ is the class associated with pixel $i$ and $p_i$ the predicted probability of a pixel with the correct class label.

### 2.2 Encoder-Decoder

A crucial step of designing an architecture is the choice of the encoder backbone since it converts the inputs to feature maps. Choosing the right neural network will result in an acceptable convergence rate. Deeper networks with smaller kernels tend to be more performant than shallower networks with larger kernels given a similar number of parameters. Although the pretrained networks VGG16, InceptionV3 and ResNet18 were initially designed for classification tasks, they can be used for segmentation purposes by serving as encoders. Moreover, transfer learning can be leveraged to significantly reduce the training time and would require fewer labeled training data. The earlier layers of a network can be fixed to help extract the main features of the new data and the rest of the layers can be retrained when dealing with a relatively small dataset. In semantic segmentation tasks, the encoder generates a tensor containing the main features such as shape and size illustrated in Fig. 2, whereas the decoder takes this information and produces the segmentation masks. Therefore, to process the generated feature maps, a decoder module is implemented. It proceeds by concatenating low level features from the early blocks of the encoder backbone with convolutional layers, dropout layers, and bilinear interpolations responsible of restoring the initial resolution of the inputs and generating the output maps. Finally, the generated dense filter maps are sent to a softmax classifier which performs pixel-wise classification.

### 2.3 Dilated Convolutions

Deep Convolutional Neural Networks (DCNNs) integrate context assimilation through continuous pooling and down-sampling layers, resulting in a loss of detail information about the object edges and degradation of the image resolution. One of the main issues of using a basic FCN architecture to perform semantic segmentation tasks is restoring the original image from a low resolution one. Thus, to perform pixel-wise labelling, the output resolution has to be increased which can be done using one of the three following methods:

- Deconvolutions: are based on creating the inverse layer of a convolutional layer. Since these deconvolution layers need to be trained, the network becomes deeper and more expensive.
Unpooling: This operation is the opposite of pooling, it consists of storing the winning activations in the different pooling layers. To restore the original input resolution, each pixel is set to the corresponding winning activation whereas its neighboring are set to 0.

Dilated convolutions: Proposed by Yu et al. [24], this approach exponentially enlarges the receptive field which results in a denser feature map. This method uses kernels of the same size as a basic convolutional layer but captures a larger field of view through the insertion of “holes” which are zero values as shown in Fig. 3. As a result, the resolution is preserved while the receptive field of the kernel is exponentially increased.

A 2D dilated convolution can be defined as follow:

$$y(m,n) = \sum_{i=1}^{M} \sum_{j=1}^{N} x(m+r, n+n+r \times j) w(i,j)$$
Where \( y(m, n) \) is the output of the dilated convolution from input \( x(m, n) \) and a filter \( w(i, j) \) with the length and the width of \( M \) and \( N \) respectively. The parameter \( r \) is the dilation rate. If \( r = 1 \), a dilated convolution turns into a normal convolution. As the dilation rate \( r \) increases, the receptive field exponentially enlarges as shown in Fig. 4:

Let \( F : \mathbb{Z}^2 \rightarrow \mathbb{R} \) be a discrete function. Let \( \Omega_r = [−r, r]^2 \cap \mathbb{R} \) be a discrete filter of size \((2r + 1)^2\). Here the discrete convolution operator \( * \) is defined as:

\[
(F * k)(p) = \sum_{s+t=p} F(s)k(t)
\]  

We now generalize this operator. Let \( l \) be a dilation factor and let \( *_l \) be defined as:

\[
(F *_l k)(p) = \sum_{s+l\cdot t=p} F(s)k(t)
\]  

We will refer to \( *_l \) as a dilated convolution or an \( l \)-dilated convolution. The familiar discrete convolution \( * \) is simply the \( 1 \)-dilated convolution. Let \( F_0, F_1, ..., F_{n-1} : \mathbb{Z}^2 \rightarrow \mathbb{R} \) be discrete functions and let \( k_0, k_1, ..., k_{n-2} : \Omega_1 \rightarrow \mathbb{R} \) be discrete \( 3 \times 3 \) filters. Let’s consider that we are applying a dilated filter with an exponential increase where: \( F_i+1 = F_i *_{2^i} k_i \) for \( i = 0, 1, ..., n-2 \) The receptive field of the element \( p \) in \( F_{i+1} \) is defined as a set of elements in \( F_0 \) that modify the value of \( F_{i+1}(p) \). Assume that the size of each element in \( F_{i+1} \) is \((2^{i+2} - 1) \times (2^{i+2} - 1) \). We implement a cascaded dilation module with geometrically increasing dilation scale (i.e. 1, 2, 4, 8). Figure 4 illustrates the cascaded module used in this work. It efficiently expands receptive field without increasing the number of parameters of the network.

![Fig. 4. Illustration of the proposed cascaded dilation module with geometrically increasing dilation scale. The final layer is a 1-dilation convolution to fuse all the former outputs.](image)

3 Experimental Study and Evaluation

3.1 Implementation Details

The experimental platform of this study is Keras [25] with Tensorflow [26] back-end, Intel Quad-Core i7-6700HQ CPU and NVIDIA GeForce GTX1060 GPU.
We used the training and validation set to estimate our hyper-parameters and avoid the adjustment of the model to the test set. For each hyper-parameter, we train the network on the training set then test it on the validation set. Subsequently, the best setup was selected to train the final model. Our network was trained for 40 epochs and a batch size of 16. The initial learning rate was set to 0.001, we used cross entropy as a loss function and Adam [27] as an optimization algorithm. Given the relatively small dataset used, over-fitting is likely to happen, therefore, to avoid over-fitting, we used a dropout rate of 50% during the training process.

3.2 Dataset

The data used in this work are 2D-RGB annotated images collected from various METU Campus building [28]. The dataset is comprised of 40000 images, 20000 for each class (Positive: images with cracks, negative: images with no cracks.) Fig. 5 shows sample images from the dataset.

![Sample images from the dataset](image)

Fig. 5. Sample images from the dataset used in this work. The top two rows illustrate images with defects whereas the bottom row has distress free images.

The resolution of the images is $227 \times 227 \times 3$. VGG16 and ResNet18 require $224 \times 224$ input images whereas InceptionV3 need $299 \times 299$ data input, therefore, we had to pre-process the images by re-scaling them in order to be accepted as inputs to the three different encoders used in this work. We split the dataset into 3 parts: 60% of the images are used for training, 20% for validation and 20% for testing. The training set was used to train the model and the verification set to observe the performance of the model during the training process. The parameters were then saved and the model was evaluated using the test set. The images included in the dataset are diversified, they are captured in different lighting conditions, have different surface roughness, color, scaling and edges.
3.3 Performance Metrics and Evaluation

In this section, we describe performance metrics used to evaluate and analyze the semantic image segmentation results. While Overall accuracy assesses the proportion of correctly labeled pixels, Per-Class accuracy represents the percent of correctly classified pixels of each individual class. When we consider Per-Class accuracy, we are essentially evaluating a binary mask; a pixel that is correctly attributed to a given class is a true positive whereas a pixel incorrectly attributed to a class is a true negative. The overall accuracy is given by:

$$Acc_{overall} = \frac{TP + TN}{TP + TN + FP + FN}$$  \hspace{1cm} (5)

where $FP$ and $FN$ represent false positives and false negatives respectively. However, this metric might lead to ambiguous results in the presence of very imbalanced class pixels in a segmented image. Therefore, we adopt the following metrics:

- **Precision**: also called positive predictive value is the fraction of relevant instances among all retrieved ones. In other words, out of all detected cracks how many are actually matching the ground truth. Precision is computed as follows:

$$Precision = \frac{TP}{TP + FP}$$  \hspace{1cm} (6)

- **Recall**: also known as sensitivity answers the following question: of all the crack annotated in the ground truth, how many were effectively captured as positive predictions in the segmented images. It is calculated as follows:

$$Recall = \frac{TP}{TP + FN}$$  \hspace{1cm} (7)

- **F-score**: measures a test’s accuracy. It is the weighted harmonic mean of the test’s precision and recall.

$$F_{score} = \left( \frac{Precision^{-1} + Recall^{-1}}{2} \right)^{-1}$$  \hspace{1cm} (8)

$$F_{score} = 2 \left( \frac{Precision . Recall}{Precision + Recall} \right)$$  \hspace{1cm} (9)

- **IoU**: also known as Jaccard index, it is used to determine how accurate is the segmentation of a given image when compared to ground truth segmentation. It is defined as follows:

$$IoU = \frac{TP}{TP + FP + FN}.$$  \hspace{1cm} (10)

- **ROC curve**: receiver operating characteristic curve is a plot that summarizes the performance of a given network at correctly classifying an object into the positive class, in our case, a crack. While the x-axis indicates the
False Positive Rate (specificity), the y-axis indicates the True Positive Rate (sensitivity). Therefore, ROC curve enables us to plot the fraction of correct predictions for the positive class versus the fraction of errors of the negative one. The best classifier has its ROC curve towards the top-left of the plot where the fraction of correct predictions is close to 1 and the fraction of incorrect negative predictions are close to 0. Specificity and sensitivity are defined as follows:

\[
\text{Specificity} = \frac{FP}{FP+TN}.
\]

\[
\text{Sensitivity} = \frac{TP}{TP+FN}.
\]

- AUC: the Area under ROC curve is usually calculated to give a single score to a model across all threshold values. It can be transcribed as the probability that the scores of a classifier will rank randomly chosen positive occurrences higher than randomly chosen negative ones.

We used VGG-16, ResNet18, and InceptionV3 as encoders in the experimental study of this work. The input resolution of VGG-16 and ResNet18 is $224 \times 224 \times 3$ whereas the input size of InceptionV3 is $299 \times 299 \times 3$. To experiment with these networks, we resize the images before feeding the data into the three models for training and testing.

Figures 6 shows that DED-InceptionV3 has a higher loss than both suggested networks: DED-VGG16 in Fig. 7 and DED-ResNet18 in Fig. 8.

We also observe that the loss of DED-VGG16 is higher than that of DED-ResNet18, but this trend becomes inverted during the test process. Moreover, the VGG-based network computation time is 2348.60 s whereas the ResNet-based network and InceptionV3-based network achieve a computation time of 3152.70 s and 8137.30 s respectively. Therefore, DED-VGG16 is faster, which can be explained by less numbers of parameters. Moreover, DED-VGG16 performed the most with an IoU of 91.78\% whereas DED-ResNet18 achieved 89.36\% IoU and DED-InceptionV3 78.26\% IoU. This also means that our proposed networks are effective at detecting fine details of the cracks since it exploits the cascaded dilation module to detect multi-scale features from multi-scale information in image inputs.

Table 1 shows the performance of the 4 semantic segmentation networks: DED-VGG16, DED-ResNet18, DED-InceptionV3 and U-Net [29]. DED-VGG16 performed the most in terms of Precision, Recall and F1-score.

Figure 9 shows the ROC curves of the different tested models and compares them to state-of-the-art model U-Net. The ROC curves of DED-VGG16, DED-ResNet 18 and U-Net on the METU dataset are comparable and the AUC of these two networks are 0.98, 0.96 and 0.92 respectively contrary to the AUC of DED-InceptionV3 that achieved 0.79. We notice that DED-VGG16 has a 2\% improvement over U-Net, in fact, DED-VGG16 has a larger receptive field which enables it to capture more details especially the thin cracks as it can be seen in the qualitative results obtained.
Fig. 6. Accuracy and loss of Training/Validation of DED-VGG16

Fig. 7. Accuracy and loss of Training/Validation of DED-InceptionV3

Fig. 8. Accuracy and loss of Training/Validation of DED-ResNet18
Table 1. Comparison of the three proposed networks and U-Net based on Precision, Recall and F1-score.

| Network            | Precision | Recall | F1-score |
|--------------------|-----------|--------|----------|
| DED-VGG16          | 0.9367    | 0.9216 | 0.9290   |
| U-Net              | 0.9135    | 0.8973 | 0.9053   |
| DED-ResNet18       | 0.8804    | 0.7958 | 0.8359   |
| DED-InceptionV3    | 0.8674    | 0.7349 | 0.7956   |

Fig. 9. ROC curves of the four compared networks: DED-VGG16, DED-ResNet18, DED-InceptionV3 and U-Net

Our results indicate that the proposed framework is efficient at performing an end-to-end crack delineation by differentiating between the background (non-cracked concrete, shadows, stains etc.) and the crack pixels. The use of CNNs encoders resulted in a robust feature extraction compared to traditional methods that consist of designing feature extractors manually. In fact, they require tuning to adapt to more complex samples, additional storage and are not suited for real-time applications. The cascaded module proposed has different dilation rates which increases the robustness of features extraction and integrate multi-scale context without reducing the resolution of the feature maps. This results in an overall improvement of the accuracy of crack delineation. The qualitative results obtained are shown in Fig. 10.
Fig. 10. Qualitative results for the proposed networks based on encoder-decoder architecture combined with a cascaded dilation module on the cracks dataset.

4 Conclusions

In this work, we present an end-to-end crack delineation framework based on an encoder-decoder meta-architecture. The model is equipped with a cascaded dilation module to enlarge the receptive fields and skip connections to fuse low-level information. First, the image samples are fed to the encoder that extracts high-level features; then, the cascaded dilation module captures context information at different scales which finally go through upsampling where they are fused with low-level features. The combination of different levels of information leads to a detailed delineation of the cracks. Although all three networks show satisfactory experimental results, DED-VGG16 achieved the best performance with a precision of 93%, recall of 92%, F1-score of 92%, and IoU of 91.78% which are higher compared to DED-InceptionV3 and DED-ResNet18. Also, the semantic segmentation qualitative results show that the framework presented is not only accurate but also robust enough to overcome interferences present in the image samples. Finally, deep Learning technics have known a tremendous amount of attention over the last few years in applications from all fields. Similarly, the civil and transportation field can benefit from these advances especially in monitoring and inspecting civil infrastructures such as bridges. As a result condition and health assessment of structures can become truly automated, cost-effective and, safe.
References

1. ARTBA analysis reveals over 47,000 U.S. bridges are structurally deficient (2019). https://roadsbridges.com/artba-analysis-reveals-over-47000-us-bridges-are-structurally-deficient. Accessed 24 Jan 2019

2. Abdel-Qader, I., Abudayyeh, O., Kelly, M.E.: Analysis of edge-detection techniques for crack identification in bridges. J. Comput. Civil Eng. 17(4), 255–263 (2003)

3. Nishikawa, T., Yoshida, J., Sugiyama, T., Fujino, Y.: Concrete crack detection by multiple sequential image filtering. Comput. Aided Civil Infrastruct. Eng. 27, 29–47 (2012). https://doi.org/10.1111/j.1467-8667.2011.00716.x

4. Dinh, T.H., Ha, Q., Lee, H.: Computer vision-based method for concrete crack detection, pp. 1–6 (2016). https://doi.org/10.1109/ICARCV.2016.7838682

5. O’Byrne, M., Ghosh, B., Schoefs, F., Pakrashi, V.: Regionally enhanced multiphase segmentation technique for damaged surfaces. Comput. Aided Civil Infrastruct. Eng. 29, 644–658 (2014). https://doi.org/10.1111/mice.12098

6. Prasanna, P., et al.: Automated crack detection on concrete bridges. IEEE Trans. Autom. Sci. Eng. 13, 1–9 (2014). https://doi.org/10.1109/TASE.2014.2354314

7. Cord, A., Chambon, S.: Automatic road defect detection by textural pattern recognition based on AdaBoost. Comput. Aided Civil Infrastruct. Eng. 27, 244–259 (2012). https://doi.org/10.1111/j.1467-8667.2011.00736.x

8. Chen, J.H., Su, M., Cao, R., Hsu, S., Lu, J.: A self organizing map optimization based image recognition and processing model for bridge crack inspection. Autom. Constr. 73, 58–66 (2017). https://doi.org/10.1016/j.autcon.2016.08.033

9. Sharma, M., Sharma, M., Leeprechanon, N., Anotaipaiboon, W., Chaiyasarn, K.: PCA Based SVM Classification Approach for Crack Detection in Concrete Structures (2016)

10. Zhang, A., Wang, K., Ji, R., Li, Q.: Efficient system of cracking-detection algorithms with 1-mm 3D-surface models and performance measures. J. Comput. Civil Eng. 30, 04016020 (2016). https://doi.org/10.1061/(ASCE)CP.1943-5487.0000581

11. Convolutional Neural Network Model Innovations for Image Classification (2019). https://machinelearningmastery.com/. Accessed 28 Feb 2020

12. Yuan, Y., Chen, X., Wang, J.: Object-Contextual Representations for Semantic Segmentation. ArXiv 2019, arXiv:1909.11065

13. Zhao, Z., Zheng, P., Xu, S., Wu, X.: Object Detection with Deep Learning: A Review. ArXiv 2019, arXiv:1807.05511v2

14. Cha, Y.J., Choi, W., Buyukozturk, O.: Deep learning-based crack damage detection using convolutional neural networks. Comput. Aided Civil Infrastruct. Eng. 32, 361–378 (2017). https://doi.org/10.1111/mice.12263

15. Fan, Z., Wu, Y., Lu, J., Li, W.: Automatic Pavement Crack Detection Based on Structured Prediction with the Convolutional Neural Network (2018)

16. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet classification with deep convolutional neural networks. Neural Inf. Process. Syst. 25, 1097–1105 (2012). https://doi.org/10.1145/3065386

17. Dorafshan, S., Thomas, R., Maguire, M.: Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. Constr. Build. Mater. 186, 1031–1045 (2018). https://doi.org/10.1016/j.conbuildmat.2018.08.011

18. LeCun, Y., Bengio, Y., Hinton, G.: Deep Learn. Nature 521, 436–44 (2015). https://doi.org/10.1038/nature14539
19. Long, J., Shelhamer, E., Darrell, T.: Fully Convolutional Networks for Semantic Segmentation. Arxiv. 2014
20. Zhang, J., Lu, C., Wang, J., Wang, L., Yue, X.: Concrete cracks detection based on FCN with dilated convolution. Appl. Sci. 9, 2686 (2019). https://doi.org/10.3390/app9132686
21. Yusiong, J.P., Naval, P.: Multi-scale Autoencoders in Autoencoder for Semantic Image Segmentation (2019). https://doi.org/10.1007/978-3-030-14799-0_51
22. Zou, Q., Zhang, Z., Li, Q., Qi, X., Wang, Q., Wang, S.: DeepCrack: learning hierarchical convolutional features for crack detection. IEEE Trans. Image Process. 28, 1 (2018). https://doi.org/10.1109/TIP.2018.2878966
23. Badrinarayanan, V., Kendall, A., Cipoll, R.: SegNet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell 39, 2481–2495 (2015). https://doi.org/10.1109/TPAMI.2016.2644615
24. Yu, F., Koltun, V.: Multi-scale Context Aggregation by Dilated Convolutions (2015)
25. Keras (2015). https://keras.io
26. Abadi, M., et al.: TensorFlow: A system for large-scale machine learning (2016)
27. Kingma, D., Ba, J.: Adam: A Method for Stochastic Optimization. International Conference on Learning Representations (2014)
28. Çağlar F. Ö.: Concrete Crack Images for Classification (2018). https://doi.org/10.17632/5y9wdsg2zt.1
29. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015, Part III. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28