Segmentation of Surface Cracks Based on a Fully Convolutional Neural Network and Gated Scale Pooling

Jacob König*† Mark David Jenkins† Peter Barrie* Mike Mannion* Gordon Morison*
*Glasgow Caledonian University, Glasgow, United Kingdom
†Geckotech Solutions Ltd., Edinburgh, United Kingdom
Email: gordon.morison@gcu.ac.uk

***Abstract*—Continual use, as well as aging, allows cracks to develop on concrete surfaces. These cracks are early indications of surface degradation. Therefore, regular inspection of surfaces is an important step in preventive maintenance, allowing reactive measures in a timely manner when cracks may impair the integrity of a structure. Automating parts of this inspection process provides the potential for improved performance and more efficient resource usage, as these inspections are usually carried out manually by trained inspectors. In this work we propose a Fully Convolutional, U-Net based, Neural Network architecture to automatically segment cracks. Conventional pooling operations in Convolutional Neural Networks are static operations that reduce the spatial size of an input, which may lead to loss of information as features are discarded. In this work we introduce and incorporate a novel pooling function into our architecture, Gated Scale Pooling. This operation aims to retain features from multiple scales as well as adapt proactively to the feature map being pooled. Training and testing of our network architecture is conducted on three different public surface crack datasets. It is shown that employing Gated Scale Pooling instead of Max Pooling achieves superior results. Furthermore, our experiments also indicate strongly competitive results when compared with other crack segmentation techniques.**

***Index Terms*—Crack Segmentation, Deep Learning, CNN, Pooling**

I. Introduction

Deep learning based methods can achieve state of the art results in many different computer vision tasks such as detection and segmentation [1]–[3]. The assessment of surface cracks in their severity is an important task. Untreated they may grow in size and critically impact the integrity of the structure, which can lead to downtime if repairs are needed and surfaces cannot be used [4]. This labour and time intensive surface inspection task is commonly carried out manually, often through a trained inspector, through either analysing images or carrying out an inspection at the target location [4], [5]. However, manual labelling of faults is very prone to human subjectivity [5].

Cracks in surfaces usually differ in color and texture from their background. These features are able to be picked up and exploited through deep learning based methods, which allows automation of this task. The task of semantic segmentation consists of labelling each pixel in an image with its corresponding class. Therefore, crack segmentation can be handled as a binary classification task, where every pixel in an image is labelled to either belong or not belong to a crack.

Many approaches to segment cracks from backgrounds have been proposed in previous literature, though the majority are non deep learning based. However, several comparisons in [6]–[11] confirm that approaches which include deep learning outperform previous conventional approaches such as thresholding [5], mathematical morphology [12] or path based methods [13], [14]. Available public data, which includes images of cracks and their segmentation masks, is sparse. To compensate for this training data deficit, available data is often augmented. This augmentation includes patch based approaches [9], [11], [15], [16] as well as flipping and cropping [6], [11].

Fully Convolutional Neural Networks (FCN) are a branch of Convolutional Neural Networks (CNN) that do not employ fully connected layers. These types of networks are often used for semantic segmentation tasks [2], [3], [6]. Deeper layers in these networks learn denser features who are then upsampled to generate segmentation masks. To combat the loss of spatial information in deeper layers, skip-connections between the down and upsampling parts were introduced in U-Net [3]. These connections allow the direct propagation of features from the encoder to the decoder part, skipping several successive layers.

Although pooling in CNN introduces rotation invariance whilst at the same time reducing the computational effort required [17], it also discards spatial information which may be important. To counteract this, several works focus on retaining and preserving these features during pooling [18]–[20].

In this paper we present a FCN architecture, based on U-Net, for crack segmentation. We introduce a new adaptable pooling function, Gated Scale Pooling (GS Pooling), which aims to retain spatial information. It learns an adaptive mixing proportion that combines Max Pooling with Stacked Pooling [20], an operation which pools features from multiple scales and merges them. This algorithm is trained and tested on the Crackforest (CFD) [21], CrackTree [22] and AigleRN [23] datasets, showing its ability to effectively segment surface cracks. We present the results using several popular metrics used in crack segmentation, ensuring that fair evaluation in future work is made possible.

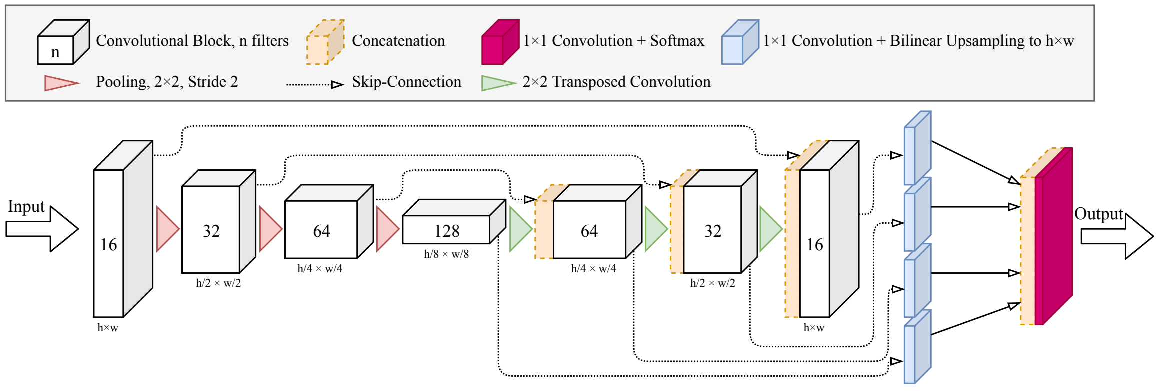

Fig. 1. Our proposed U-Net based architecture for crack segmentation. Pooling uses our proposed Gated Scale Pooling operation. The number of filters used during 1 × 1 convolutions is equivalent to the number of output classes, two in this binary crack segmentation case.

II. Approach

A. U-Net

This work makes use of an architecture which is modelled following U-Net [3]. It consists of an encoder and a decoder part, that are connected through shortcut connections. The encoder part follows the common architecture of CNN, using successions of convolutional, activation and pooling layers. Every pooling layer in this architecture reduces the spatial dimensions of the feature map by a factor of two. The decoder part of this architecture mirrors the encoder part. However, instead of using pooling operations, transposed convolutions are used which upscale the feature map by a factor of two. Through use of the shortcut connections between the encoder and decoder, spatial information is retained in deeper layers which would otherwise be lost due to the pooling operation. An illustration of our architecture is shown in Fig. 1.

In detail, this architecture makes use of seven convolutional blocks, four in the encoder and three in the decoder section. In the encoder section these blocks are connected through three pooling operations, after which the number of filters in each block is doubled. In the decoder path, after each upsampling operation, the number of filters is halved. Furthermore, the input into each decoder convolutional block consists of the concatenation of the opposite convolutional block in the encoder section, as well as the upsampled feature map from the previous block.

To allow the features extracted at each scale to influence the segmentation output we employ a multi scale fusion, following the Deep Supervision approach [24], [25]. This means that the output of the last convolutional block of the encoder part, as well as the output from all the convolutional blocks in the decoder part are each being fed through a 1×1 linear transformation with two filters. These four feature maps are then upsampled using bilinear interpolation, to match the spatial dimensions of the input. This is followed by concatenating these feature maps along the channel dimension and applying another 1×1 linear transformation with two filters. To obtain the final segmentation map through this feature map we then apply a Softmax activation function, whose output represents the confidence scores whether a pixel belongs to a crack or not.

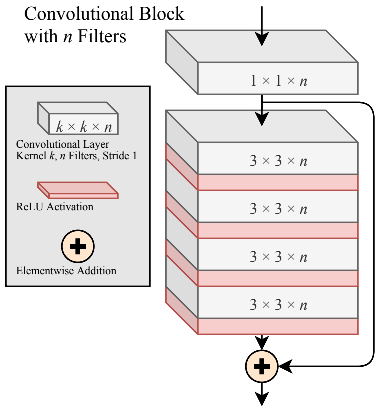

Fig. 2. Convolutional block in this architecture.

Each convolutional block features a 1×1 linear transformation, four 3×3 convolutions, followed by ReLu activation functions, as well as a residual connection [26] as proposed in the architecture in [27]. This is illustrated in Fig. 2.

The convolutional layers in this architecture make use of partial convolution based padding [28]. This approach weighs output features next to borders during a convolution, based on the ratio of zero padded features to the ratio of features on the kernel position during the sliding window operation. It aims to improve convergence as well as improve performance of the network, compared to using zero padding.

B. Gated Scale Pooling

In this work we introduce Gated Scale Pooling . This pooling operation is based on Gated Max-Average [19] and Stacked Pooling [20]. Stacked Pooling is used to provide scale invariance by subsequently max-pooling a feature map using differently sized kernels and strides, followed by extracting an elementwise average from the output of each pooling operation. The Gated Max-Average Pooling learns a gating mask which combines the Average and Max Pooling operation.

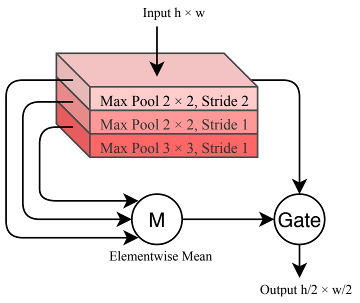

Fig. 3. Gated Scale Pooling operation, combining Stacked Pooling (all three blocks) with conventional Max Pooling (first block). The Gate Operation uses a gating mask on the input, followed by squashing the result through a Sigmoid activation function. This gating operation then combines the output of those two pooling operation on a per-channel basis.

Expanding upon that, we propose to include Stacked Pooling instead of Average Pooling, thus creating GS Pooling. The aim of GS Pooling is to automatically adapt to the features being pooled, whilst at the same time providing scale insensitivity. Due to making use of an adaptive gate, which learns the parameters during training, the optimal mixing proportion of Max Pooling and Stacked Pooling is chosen. During training and inference the gating mask gates the input features on a per-channel basis, which is equivalent to applying a depthwise convolution with one filter per channel on the input. This gating mask output is then put through a Sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ to squash its values into a range of [0,1]. Let x be the input into the pooling operation p. Generating the pooled output can then be denoted as:

$$p(x) = M(x)\sigma(g(x)) + S(x)(1 - \sigma(g(x))) \tag{1}$$

with M, S, g representing the Max Pooling, Stacked Pooling and gating operation respectively. This pooling operation is illustrated more clearly in Fig. 3, where the first pool block represents the conventional Max Pooling operation and all three of these pool blocks and their combination through the elementwise-mean represent Stacked Pooling.

III. Experiments and Results

A. Metrics

In previous works on Crack Segmentation, multiple different metrics are used to compare the performance of algorithms. The works in [9], [16], [21], [22] make use of F1-Score $F1$, Recall RE and Precision PR for a fixed confidence threshold. However, we report our results following the metrics in [6], [7], [29]: $F1$ on a fixed threshold ODS (Optimal Dataset Scale), Recall RE_{ODS} and Precision PR_{ODS} on this fixed threshold, as well as the average $F1$ for the best threshold on each image OIS (Optimal Image Scale), calculated through using all confidence thresholds $t \in [0.01, 0.99]$ with intervals of 0.01. These metrics are calculated as follows:

$$ODS = max\Big\{F1_t : \forall t \in \{0.01, ..., 0.99\}\Big\} \tag{2}$$

$$OIS = \frac{1}{N_{img}} \sum_{i}^{N_{img}} max\Big\{F1_t^i : \forall t \in \{0.01, ..., 0.99\}\Big\} \tag{3}$$

with N_{img} representing the total number of images on which evaluation is run.

As in [6], [16] we consider a pixel being correctly classified if the prediction lies withing a threshold of two pixels to a corresponding ground truth pixel.

B. Datasets

To investigate the ability of this architecture to generalize well across multiple datasets, it is trained and tested on three different datasets:

- *Crackforest* (CFD) [21] This dataset consists of 117 images of size 480×320 pixels (one image was discarded as the ground truth was incorrect). The ratio of crack to non-crack pixels is 1:61.
- *CrackTree* [22] It contains 206 images of size 800×600 pixels. These images also extensively include shadows and low contrast. The ratio of crack to non-crack pixels is 1:312.
- *AigleRN* [23] AigleRN is a small dataset consisting of 38 images. Half of these images are of size 991×462 whilst the other half is of size 311×462. The ratio of crack to non-crack pixels is 1:139.

All of these datasets include their annotated binary ground truth segmentation map.

C. Implementation and Training

For comparison purposes we follow the train/test split for CFD and AigleRN proposed in [16]. CFD is split into 71 train and 46 test images, whereas AigleRN is split into 24 train and 14 test images. Our proposed split for CrackTree consists of 130 train and 76 test images.

We employ a patch training and testing process, using a patch size of 48 × 48 pixels. During training, patches are randomly extracted from each image using a ratio of 60% patches, that contain at least one crack pixel, to 40% patches without crack pixels. For each image in CFD and CrackTree we extract 2000 and 3000 patches respectively. We utilize a dynamic extraction approach in AigleRN, as it contains differently sized images and the ratio of cracks to non crack pixels is much higher than in the two other datasets. In this approach, the number of patches to extract from each image is chosen based on the total number of crack pixels in one specific image relatively to the total amount of crack pixels in the training dataset. Therefore, more training patches are extracted from images whose total number of crack pixels is higher. The total number of patches to extract from this dataset is set to 72,000. Image preprocessing before training on all datasets is done using histogram equalization, normalization and gamma adjustment.

On each dataset the architectures are trained for 25 epochs. During training SGD is applied for optimization, utilising a learning rate of 0.001 and a momentum of 0.9. The loss function is chosen to be sum of Binary Cross Entropy and Dice Loss [30] and the batch size is set to 32.

TABLE I
RESULTS ON CFD.

Method	OIS	ODS	PR_{ODS}	RE_{ODS}
Ours, GS Pooling	**95.84**%	**94.92**%	95.84%	94.02%
Ours, Max Pooling	95.72%	94.76%	**95.89**%	93.66%
CNN [16]	-	92.44%	91.19%	**94.81**%
Crackforest [21]	-	85.71%	82.28%	89.44%

TABLE II
RESULTS ON CRACKTREE.

Method	OIS	ODS	PR_{ODS}	RE_{ODS}
Ours, GS Pooling	**88.60**%	**87.63**%	**88.93**%	**86.37**%
Ours, Max Pooling	88.32%	87.01%	87.84%	86.19%

TABLE III
RESULTS ON AIGLERN.

Method	OIS	ODS	PR_{ODS}	RE_{ODS}
Ours, GS Pooling	**90.64**%	**90.24**%	89.33%	91.17%
Ours, Max Pooling	89.75%	89.05%	86.89%	**91.31**%
CNN [16]	-	89.54%	**91.78**%	88.12%

D. Results

To evaluate the performance of our architecture with GS Pooling we implement two models based on the previously described architecture: one including GS Pooling as the pooling operation (*Ours, GS Pooling*) and one that uses Max Pooling as the pooling operation (*Ours, Max Pooling*). The patch based testing process extracts a patch at every possible position in an image, utilising a sliding window with a stride of 1 in the height and width dimension. After feeding these patches through the trained network, the output segmentation map is generated by averaging all prediction results at each possible pixel position in each image. Table I, Table II and Table III present the results on the CFD, CrackTree and AigleRN datasets respectively.

The results on all datasets indicate that making use of GS Pooling instead of Max Pooling achieves a higher performance, ranging from 0.89% and 1.19% in OIS and ODS on AigleRN to 0.29% and 0.61% on CrackTree as well as a slight improvement of 0.13% in OIS and 0.16% in ODS on CFD. Furthermore, the results in Table I and Table III show that this architecture outperforms previous state of the art results in those datasets on the ODS by 2.48% on CFD metric and 0.7% on AigleRN. However, it is to note that the competing CNN method [16] did not provide source code, therefore we make use of the results reported in their work. In addition to that, we assume that the results of the competing CNN [16] as well as the Crackforest [21] method have been generated using the best possible confidence threshold, therefore we report them under the ODS metric. Fig. 4 shows the crack segmentation results on a sample image from each dataset. As it can be seen, the majority of cracks in images are segmented. However, as seen in the predictions, the algorithm may interpret some noise as a crack, especially when the color is similar to that of cracks.

IV. CONCLUSION

Detecting and labelling surface cracks is a task which benefits from automation through computer vision methods. In this work we presented an U-Net based Convolutional Neural Network architecture for semantic segmentation of road cracks. In addition to that we also introduced a novel pooling function, Gated Scale Pooling. This pooling function aims to retain relevant spatial information from multiple scales through combining two pooling operations, Max Pooling and Stacked Pooling, using an adaptive mixing proportion. We employ an image-patch based training and testing process, training and evaluating this model on three datasets. Our results indicate that our architecture incorporating Gated Scale Pooling outperforms the same architecture with Max Pooling in all three of the datasets. Furthermore, we obtain new state of the art results in the two datasets, where results from other works were available. We observe that the results of this model achieve satisfactory segmentation results on all three datasets, indicating that this architecture generalizes well.

In the future we aim to study how augmentation as well as different patch sizes affect the results on crack segmentation. Moreover, we plan on conducting experiments studying the effects of utilizing Gated Scale Pooling in other network architectures and domains.

ACKNOWLEDGMENT

The work presented in this paper has been carried out in association with Geckotech Solutions Ltd. who provide specialist access solutions to the railway and construction engineering industries. The authors would like to thank Geckotech Solutions Ltd. for their support. The authors would also like to thank The Data Lab for their funding of the Industrial Doctorate.

REFERENCES

[1] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "Hybrid Task Cascade for Instance Segmentation," *arXiv preprint arXiv:1901.07518*, Jan. 2019.

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[4] 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 *International Joint Conference on Neural Networks*. IEEE, 2017, pp. 2039–2047.

[5] H. Oliveira and P. L. Correia, "Automatic Road Crack Segmentation Using Entropy and Image Dynamic Thresholding," in *17th European Signal Processing Conference*. IEEE, 2009, pp. 622–626.

[6] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang, "DeepCrack: Learning Hierarchical Convolutional Features for Crack Detection," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1498–1512, 2019.

[7] F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, and H. Ling, "Feature Pyramid and Hierarchical Boosting Network for Pavement Crack Detection," *arXiv preprint arXiv:1901.06340*, Jan. 2019.

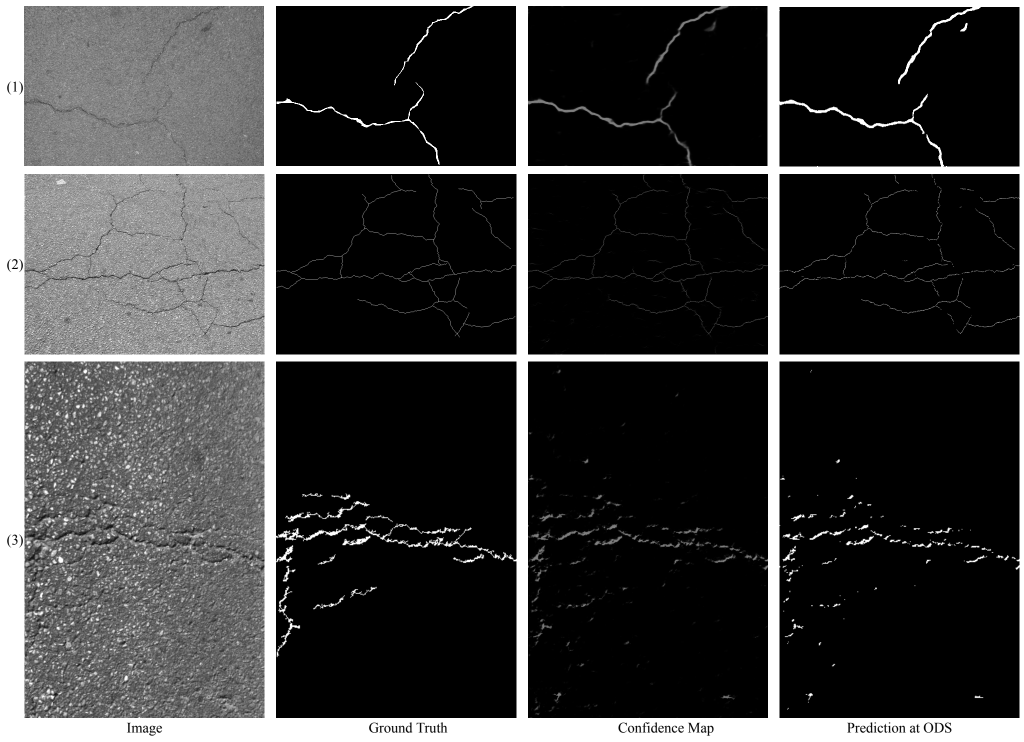

Fig. 4. Evaluation results on CFD (1), CrackTree (2) and AigleRN (3). The rightmost predictions were extracted at the confidence threshold from the ODS result.

[8] N.-D. Hoang, Q.-L. Nguyen, and V.-D. Tran, "Automatic Recognition of Asphalt Pavement Cracks using Metaheuristic Optimized Edge Detection Algorithms and Convolution Neural Network," *Automation in Construction*, vol. 94, pp. 203–213, 2018.

[9] M. D. Jenkins, T. A. Carr, M. I. Iglesias, T. Buggy, and G. Morison, "A Deep Convolutional Neural Network for Semantic Pixel-Wise Segmentation of Road and Pavement Surface Cracks," in *26th European Signal Processing Conference*, 2018, pp. 2120–2124.

[10] K. Gopalakrishnan, S. K. Khaitan, A. Choudhary, and A. Agrawal, "Deep Convolutional Neural Networks with Transfer Learning for Computer Vision-Based Data-Driven Pavement Distress Detection," *Construction and Building Materials*, vol. 157, pp. 322–330, 2017.

[11] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road Crack Detection using Deep Convolutional Neural Network," in *IEEE International Conference on Image Processing*. IEEE, 2016, pp. 3708–3712.

[12] N. Tanaka and K. Uematsu, "A Crack Detection Method in Road Surface Images Using Morphology," *Proceedings of the Workshop on Machine Vision Applications*, pp. 154–157, 1998.

[13] V. Kaul, A. Yezzi, and Y. Tsai, "Detecting Curves with Unknown Endpoints and Arbitrary Topology Using Minimal Paths," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 10, pp. 1952–1965, Oct. 2012.

[14] 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 Transactions on Intelligent Transportation Systems*, vol. 17, no. 10, pp. 2718–2729, 2016.

[15] T. A. Carr, M. D. Jenkins, M. I. Iglesias, T. Buggy, and G. Morison, "Road Crack Detection using a Single Stage Detector Based Deep Neural Network," in *IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems*. IEEE, 2018, pp. 1–5.

[16] Z. Fan, Y. Wu, J. Lu, and W. Li, "Automatic Pavement Crack Detection Based on Structured Prediction with the Convolutional Neural Network," *arXiv preprint arXiv:1802.02208*, 2018.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[18] F. Saeedan, N. Weber, M. Goesele, and S. Roth, "Detail-Preserving Pooling in Deep Networks," *arXiv preprint arXiv:11804.04076*, Apr. 2018.

[19] C. Y. Lee, P. Gallagher, and Z. Tu, "Generalizing Pooling Functions in CNNs: Mixed, Gated, and Tree," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 863–875, Apr. 2018.

[20] S. Huang, X. Li, Z.-Q. Cheng, Z. Zhang, and A. Hauptmann, "Stacked Pooling: Improving Crowd Counting by Boosting Scale Invariance," *arXiv preprint arXiv:1808.07456*, 2018.

[21] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen, "Automatic Road Crack Detection using Random Structured Forests," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3434–3445, 2016.

[22] Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang, "CrackTree: Automatic Crack Detection from Pavement Images," *Pattern Recognition Letters*, vol. 33, no. 3, pp. 227–238, 2012.

[23] S. Chambon and J.-M. Moliard, "Automatic Road Pavement Assessment with Image Processing: Review and Comparison," *International Journal of Geophysics*, vol. 2011, 2011.

[24] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-Supervised Nets," in *Artificial Intelligence and Statistics*, 2015, pp. 562–570.

[25] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz, and Others, "Attention U-Net: Learning Where to Look for the Pancreas," *arXiv preprint arXiv:1804.03999*, 2018.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[27] M. Z. Alom, M. Hasan, C. Yakopcic, T. M. Taha, and V. K. Asari, "Recurrent Residual Convolutional Neural Network based on U-Net (R2U-Net) for Medical Image Segmentation," *arXiv preprint arXiv:1802.06955*, 2018.

[28] G. Liu, K. J. Shih, T.-C. Wang, F. A. Reda, K. Sapra, Z. Yu, A. Tao, and B. Catanzaro, "Partial Convolution based Padding," *arXiv preprint arXiv:1808.07456*, Nov. 2018.

[29] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour Detection and Hierarchical Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, May 2011.

[30] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation," in *International Conference on 3D Vision*. IEEE, 2016, pp. 565–571.