JOURNAL OF THE CIVIL ENGINEERING FORUM

Journal of the Civil Engineering Forum, September 2021, 7(3): 323-334
DOI 10.22146/jcef.65288
Available Online at http: https://jurnal.ugm.ac.id/jcef/issue/archive

Crack Detection on Concrete Surfaces Using Deep Encoder-Decoder Convolutional Neural Network: A Comparison Study Between U-Net and DeepLabV3+

Patrick Nicholas Hadinata[1,*], Djoni Simanta[1], Liyanto Eddy[1], Kohei Nagai[2]
[1]Department of Civil Engineering, Universitas Katolik Parahyangan, Bandung, INDONESIA
Jalan Ciumbuleuit No 94 Bandung
[2]Institute of Industrial Science, The University of Tokyo, Tokyo, JAPAN
4-6-1 Komaba Meguro-ku Tokyo-to
[*]Corresponding authors: pnhadinata@gmail.com

SUBMITTED 14 April 2021 **REVISED** 10 May 2021 **ACCEPTED** 31 May 2021

ABSTRACT Maintenance of infrastructures is a crucial activity to ensure safety using crack detection methods on concrete structures. However, most practice of crack detection is carried out manually, which is unsafe, highly subjective, and time-consuming. Therefore, a more accurate and efficient system needs to be implemented using artificial intelligence. Convolutional neural network (CNN), a subset of artificial intelligence, is used to detect cracks on concrete surfaces through semantic image segmentation. The purpose of this research is to compare the effectiveness of cutting-edge encoder-decoder architectures in detecting cracks on concrete surfaces using U-Net and DeepLabV3+ architectures with potential in biomedical, and sparse multiscale image segmentations, respectively. Neural networks were trained using cloud computing with a high-performance Graphics Processing Unit NVIDIA Tesla V100 and 27.4 GB of RAM. This study used internal and external data. Internal data consisted of simple cracks and were used as the training and validation data. Meanwhile, external data consisted of more complex cracks, which were used for further testing. Both architectures were compared based on four evaluation metrics in terms of accuracy, F1, precision, and recall. U-Net achieved segmentation accuracy = 96.57%, F1 = 87.55%, precision = 88.15%, and recall = 88.94%, while DeepLabV3+ achieved segmentation accuracy = 96.47%, F1 = 85.29%, precision = 92.07%, and recall = 81.84%. Experiment results (internal and external data) indicated that both architectures were accurate and effective in segmenting cracks. Additionally, U-Net and DeepLabV3+ exceeded the performance of previously tested architecture, namely FCN.

KEYWORDS Convolutional Neural Network; U-Net; DeepLabV3+; Crack Detection; Maintenance of Infrastructures.

1 INTRODUCTION

The periodic maintenance of infrastructure is necessary to prevent deterioration, guarantee operational safety and prolonged lifespan. One of the commonly occurring types of structural damages is crack on concrete surfaces. There are various causes, a couple of which are fatigue and overloading. Crack detection on concrete surfaces is a key step in guaranteeing the health of the infrastructure. Nowadays, this practice mostly relies on manual inspection and professional assessment. This method is highly expensive, work-intensive, time-consuming, risky, and biased. Therefore, a more equitable and measurable approach can be implemented using computer vision techniques.

In recent years, artificial intelligence has undergone extensive research and development, including computer vision-related issues. The main component is an artificial neural network (ANN). It is an engineered system that imitates a biological brain. A convolutional neural network (CNN) is a subset of ANN specifically designed to offer a sense of vision. In addition, CNN tremendously contributes to the development of computer vision techniques primarily used in detecting and classifying objects. It helps to analyze the complex spatial mathematical descriptions of images. Complex features are mathematically segmented by a CNN technique called convolution.

Semantic image segmentation is a CNN application used to categorize each pixel to a specific class. CNN is structured as an encoder-decoder to perform semantic image segmentation. The encoder extracts features from an image used to generate a feature map, which is downsampled with various techniques used to obtain salient information and reduce computation cost. The decoder assigns labels to the feature map and further upsamples the image to its original dimension. In CNN, downsampling and upsampling are both image resampling processes for dimensional reduction and enlargement, respectively. Both processes need to maintain salient information. Figure 1 shows a simple illustration of an encoder-decoder basic architecture.

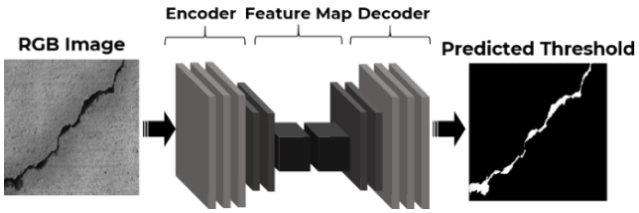

Figure 1. Encoder-decoder network.

Fully Convolutional Network (FCN), introduced by Long *et al.* (2015), has been the base architecture for many cutting-edge semantic image segmentation networks. The bottom line of FCN is to replace the fully connected layers with the convolutional ones, structured into an encoder-decoder which yields a pixel-wise classification with the same dimensions as the input. Afterward, numerous architectures were developed from FCN, for example, SegNet, PSPNet, U-Net, and DeepLabV3+. U-Net was introduced by Ronneberger *et al.* (2015) as an efficient and flexible architecture mainly used for segmenting biomedical images. Similarly, it also provides an encoder-decoder structure with some modifications and extensions. U-Net introduces the concatenate processes to combine the feature maps from the encoder to the decoder. These processes are carried out in a multi-level fashion to assemble a precise segmentation from the decoder module. Furthermore, DeepLabV3+ (Chen *et al.*, 2018) was introduced with modified Xception backbone (Chollet, 2017) and Atrous Spatial Pyramid Pooling (ASPP) module as its main features. This led to its computational efficiency and rapid convergence. DeepLabV3+ was tested with PASCAL VOC 2012 dataset to prove its effectiveness, ultimately reaching an mIoU of 89%.

Yang *et al.* (2018) tested FCN for crack detection, which conclusively reached a training F1 score of 79.95%. However, FCN often ignores detailed information, which results in a smoothened segmentation output. Both U-Net and DeepLabV3+ have been proven to be more powerful and accurate compared to FCN. Therefore, this study is aimed to compare the effectiveness of U-Net and DeepLabV3+ on crack segmentation. Both models are tested using internal and external data.

2 METHODS

2.1 Non-linear Activation Functions

ReLu and Sigmoid serve as non-linear activation functions in neural networks. Initially, Sigmoid was widely used on every layer. However, based on vanishing gradients and costly computation, Nair and Hinton (2010) introduced ReLu. ReLu provides a rapid convergence as it eliminates impractical negative values in the image recognition tasks. ReLu outputs 0 for a negative input and a simple linear function for positive input. Furthermore, ReLu effectively produces only 0 and 1 as gradients, eliminating the vanishing gradient problem and boosting computation efficiency for backpropagation. Sigmoid provides a non-linear exponential-based function for binary classification. It yields a real value between 0 and 1 to generate a probability for the 2 classes. ReLu is utilized in the middle layers, while Sigmoid is used in the final layer used for binary prediction output. ReLu and Sigmoid are defined in Equations (1) and (2), respectively. Moreover, Figures 2 and 3 show the ReLu and Sigmoid functions.

$$g(z) = \begin{cases} 0, & z < 0 \\ z, & z \geq 0 \end{cases} \quad (1)$$

$$g(z) = \frac{1}{1+e^{-z}} \quad (2)$$

where z is the input to a neuron.

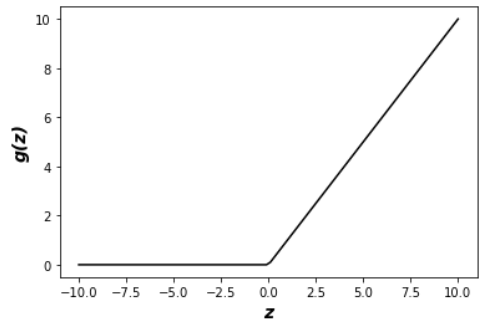

Figure 2. ReLu function.

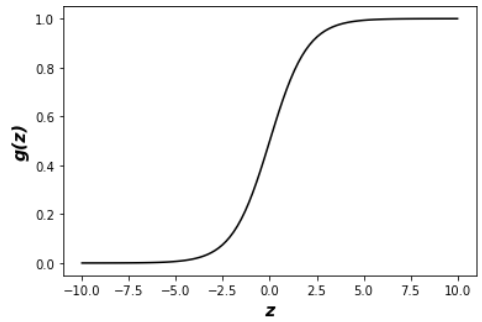

Figure 3. Sigmoid function.

2.2 U-Net Architecture

U-Net was originally introduced to segment biomedical images, especially for living cells. It won the ISBI cell tracking challenge held in 2015 by a large margin. Subsequently, U-Net has been slightly modified in many ways for a rapid and accurate custom segmentation task. Besides, it possesses a symmetrical "U" shaped architecture, as the name suggests. Its encoder comprises convolution and max-pooling operations. The number of channels is doubled from 64 in the earliest layer to 1024, eventually in the bottleneck layer. The decoder upsamples the extracted feature map and convolves it with the same number of channels as the encoder. In addition, feature maps from the encoder are linked to the decoder using skip connections which aims to retain information and recover fine-grained details, leading to precise object localization. The overall U-Net architecture, as shown in Figure 4, contains 31 million trainable parameters.

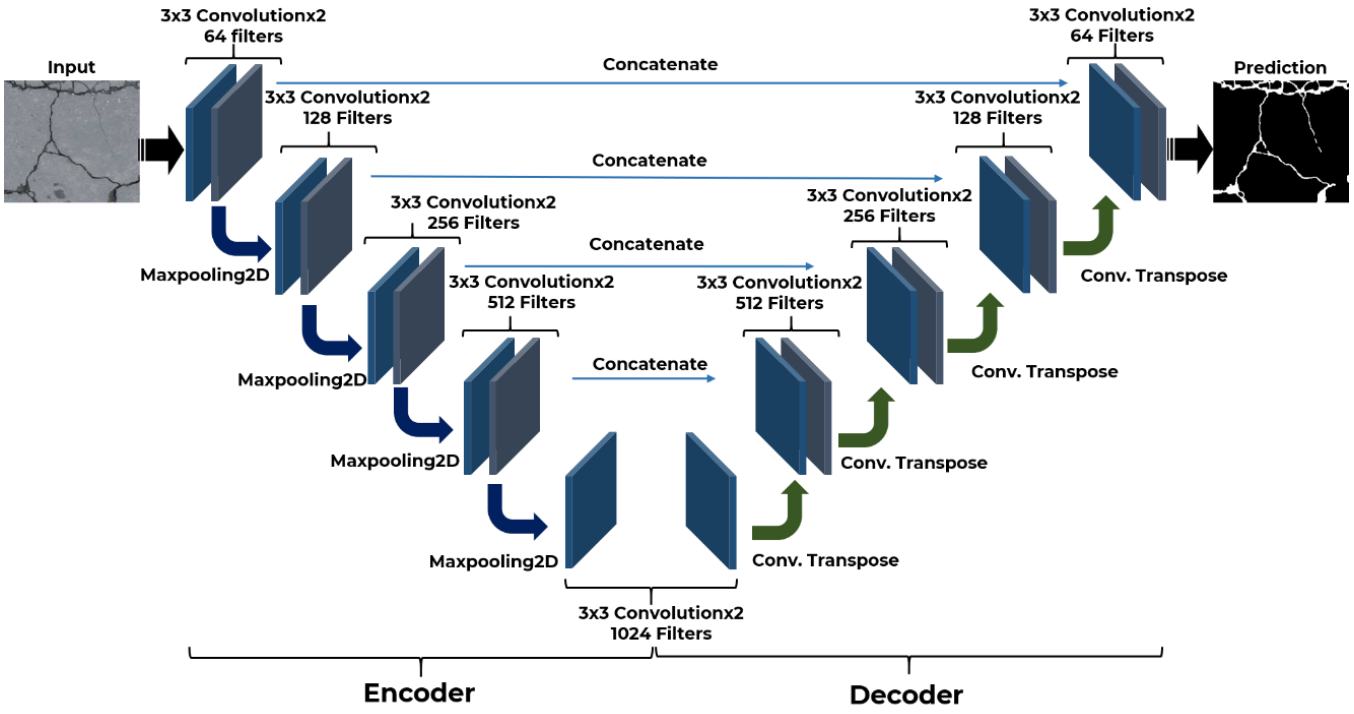

Figure 4. U-Net architecture (Ronneberger *et al.*, 2015).

2.3 DeepLabV3+ Architecture

DeepLabV3+ is an advancement of the former naïve decoder architecture, namely DeepLabV3 (Chen *et al.*, 2017). It aims to provide a deep multiscale spatial information of images using a feature extractor consisting of a modified Xception backbone and an ASPP module. The xception backbone was modified by Chen *et al.* (2018) by adding more layers, changing the max-pooling operations with strided depthwise separable convolutions (Chollet, 2017), and adding extra batch normalization (Ioffe & Szegedy, 2015). ReLu followed this after every depthwise convolution. The ASPP module consists of atrous depthwise separable convolutions with different rates (6, 12, 18), global average pooling, and 1×1 convolution. Moreover, the ASPP layers are linked together by a concatenate process to combine the extracted multiscale contextual information. An output stride of 16 is used for this segmentation task. Therefore, the last 2 blocks' striding in the modified Xception backbone is replaced with dilated convolutions. The overall DeepLabV3+ architecture contains 47 million trainable parameters. DeepLabV3+ and modified Xception backbone details are shown in Figures 5 and 6, respectively.

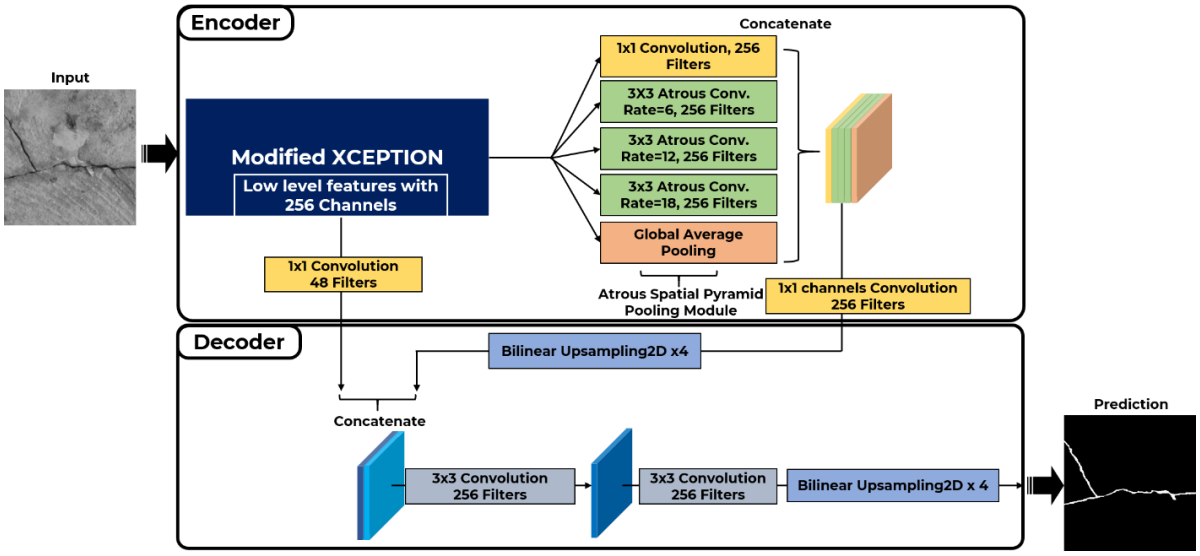

Figure 5. DeepLabV3+ architecture (Chen *et al.*, 2018).

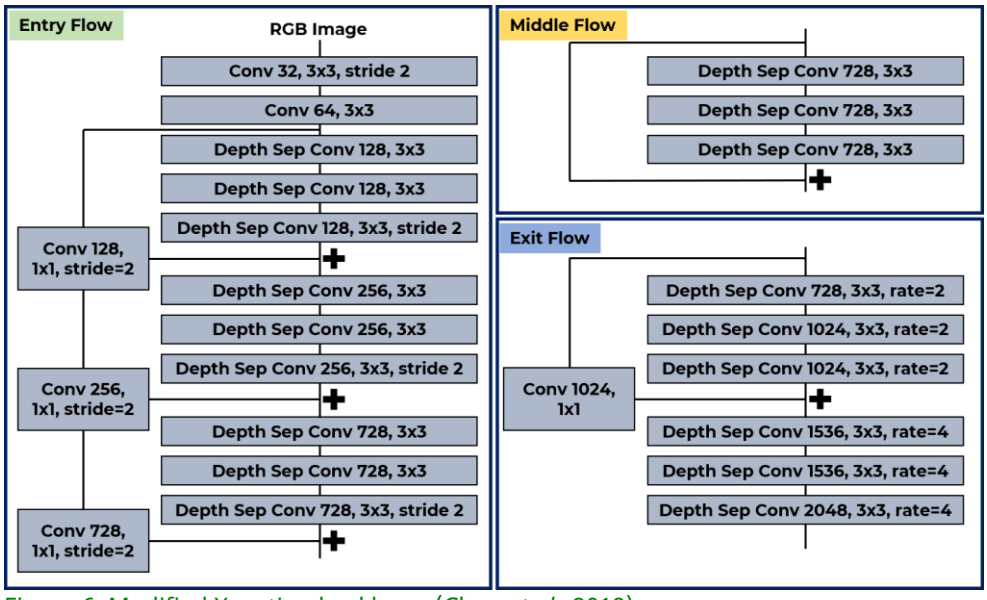

Figure 6. Modified Xception backbone (Chen *et al.*, 2018).

2.4 Adam Optimizer

Semantic image segmentation contains sparse amount of data, thereby leading to a highly non-convex loss contour. This typically hides numerous local minima and plateau, which makes it difficult for neural networks to converge. Kingma and Ba (2014) reported that adaptive moment estimation (Adam) optimizer applies adaptive effective learning rates to escape these obstacles. Adam keeps the exponential moving average of the past squared gradient similar to RMSprop, as well as the exponential moving average of the past gradient similar to SGD with momentum (Qian, 1999). Adam parameters update rule are represented in Equations (3) to (8).

$$g_j = -\nabla L \tag{3}$$

$$s_j^i = \rho_1 s_j^{i-1} + (1-\rho_1) g_j \tag{4}$$

$$r_j^i = \rho_2 r_j^{i-1} + (1-\rho_2)\left(g_j\right)^2 \tag{5}$$

$$\hat{s}_j^i = \frac{s_j^i}{1+\rho_1} \tag{6}$$

$$\hat{r}_j^i = \frac{r_j^i}{1+\rho_2} \tag{7}$$

$$W_j^{i+1} = W_j^i + \frac{\epsilon \hat{s}_j}{\delta + \sqrt{\hat{r}_j}} \tag{8}$$

where g_j is the observed gradient of specific learning step, s_j is the update for the first moment estimate, r_j is the update for the second moment estimate, $\hat{s}_j$ is the bias correction for the first moment estimate, $\hat{r}_j$ is the bias correction for the second moment estimate, W_j is the weight update, ρ_1 is a hyperparameter with a default value of 0.9 containing the exponential decay of the first moment, ρ_2 is a hyperparameter with a default value of 0.999 containing the exponential decay value of the second moment, δ is a constant to prevent zero denominators with a default value of 10^{-7}, and ε is the learning rate. Adam has been assessed in various image recognition tasks and proved to be one of the most efficient optimizers.

2.5 Evaluation Metrics

Evaluation of binary image segmentation requires 4 metrics variables, namely true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Figure 7 shows that TP indicates crack prediction where the label is also crack, FP indicates crack prediction where the label is background, FN indicates background prediction where the label is crack, and TN indicates background prediction where the label is also background. TP, TN, FP, and FN metrics were expanded into accuracy, precision, recall, and F1, which are stated in Equations (9), (10), (11), and (12), respectively.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{9}$$

$$Precision = \frac{TP}{TP+FP} \tag{10}$$

$$Recall = \frac{TP}{TP+FN} \tag{11}$$

$$F1 = \frac{2\times TP+\gamma}{2\times TP+FN+FP+\gamma} \tag{12}$$

		Prediction	
		Crack	Background
Groundtruth	Crack	TP	FN
	Background	FP	TN

Figure 7. Confusion matrix of evaluation variables.

Accuracy signifies the correct overall prediction for both positive and negative values. According to Goutte and Gaussier (2005), precision signifies the total of correct positive values divided by the total positive prediction. Recall signifies the total of correct positive values divided by the total of positive labels. Models cannot be compared only by precision and recall metrics since both metrics interpret different variables. Therefore, a balanced combination of precision and recall metrics needs to be defined, namely F1. F1 harmonically averages both precision and recall values, which is a more significant benchmarking metric for a highly imbalanced segmentation. Therefore, it is the deciding metric to compare both architectures.

2.6 Dice Loss Function

A binary classification, namely cracks (labeled as 1) and non-crack (labeled as 0), is used. Cracks tend to be much smaller compared to the background of the image. Therefore, it is arguably a highly unbalanced segmentation task. Loss function with the foreground defined as an equally weighted variable to the positive labels, such as binary cross-entropy, tend to converge slowly, which strongly correlates to the local minima. Milletari *et al.* (2016) introduced the dice loss function in such a way that it computes F1 as the only variable that needs to be enhanced. Based on further testing, it was concluded that this function converges faster than the binary cross-entropy. The whole dice loss function is defined in Equation (13).

$$L = 1 - F1\big(P(y,\hat{y}), R(y,\hat{y})\big) = 1 - \frac{2\times TP+\gamma}{2\times TP+FN+FP+\gamma} \quad (13)$$

where P and R are the precision and recall values, respectively, and γ is the smoothening factor to prevent an undefined function when the denominator is 0. For practicality, γ is set as 1 in this study.

2.7 Dropout Regularization

Srivastava *et al.* (2014) stated that dropout regularizes the neural network by averaging weights across every epoch. As shown in Figure 8, x_n is the input to a neural network, $a_n^{(l)}$ is the value generated by a neuron of layer l, $b^{(l)}$ is the bias of layer l, and y_{pred} is the output of a neural network. Dropout randomly deactivates a defined proportion of a_n in a single layer of neural networks. This averaging effect prevents neural networks to fall into high variance fitting caused by an arguably large weight in a large network. A dropout of 0.2 randomly deactivates 0.2 nodes of a single layer. Dropout was configured with values ranging from 0.1 to 0.3 both in the encoder and decoder section. Based on experiment, it was concluded that a higher dropout rate in the deeper layer resulted in a better segmentation output. The dropout rate of 0.1 and 0.2 was set in the first and middle blocks of the U-Net, with 0.3 in the bottleneck layer. The same rate was also applied in the decoder of U-Net. The dropout rate of 0.3 was set in the last block of the modified Xception encoder of DeepLabV3+. Meanwhile, the decoder section receives a dropout rate of 0.2. Every dropout layer is set after that of the convolutional ones.

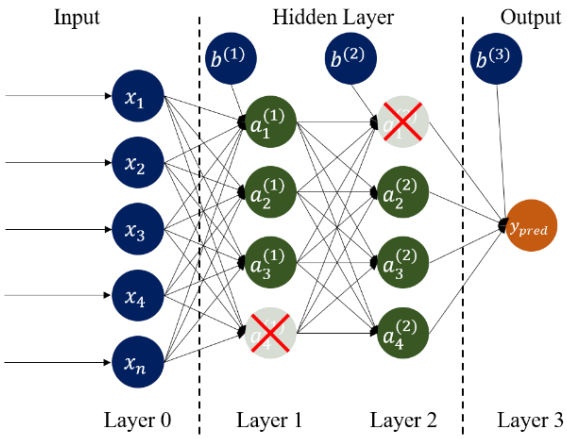

Figure 8. Neural networks after applying dropout.

2.8 Fitting Hyperparameters

The learning rates of U-Net and DeepLabV3+ are set at 4×10^{-4} and 2×10^{-4}, with 8 and 4 as the batch sizes, respectively. Both parameters are set in such a way that a stable learning curve with the best result is obtained for each network respectively.

2.9 Validation Data

Validation data is sourced from the training dataset, which is shuffled randomly to prevent poor generalization. The validation proportion is set to 0.2, with every proportion cross-validated to ensure the best generalization. A validation proportion of 0.3 leads to underfitting, while 0.1 produces an unstable validation curve. This causes a highly biased result and cannot be interpreted as an objective comparison.

2.10 Dataset

An internal crack dataset used for training and validation was obtained from the study carried out by Yang *et al.* (2018). It is comprised of 776 crack images with corresponding ground-truth segmentation. Images have a dpi ranging from 72 to 300 with different environmental scenarios and color schemes. Some examples of the internal dataset are shown in Figure 9. To further test

neural networks' ability for crack segmentation, external data consisting of images with complex scenarios and surface features, were collected. Figure 10 shows examples of external data.

All images are preprocessed before being fed into the neural network for training and testing. The preprocessing stage is divided into 2 parts. Firstly, all images are bilinearly resized to 256×256 pixels. Secondly, they are normalized by dividing each pixel value by 255. This reduces the computational cost and provides better convergence for binary output. No further preprocessing was carried out in this study.

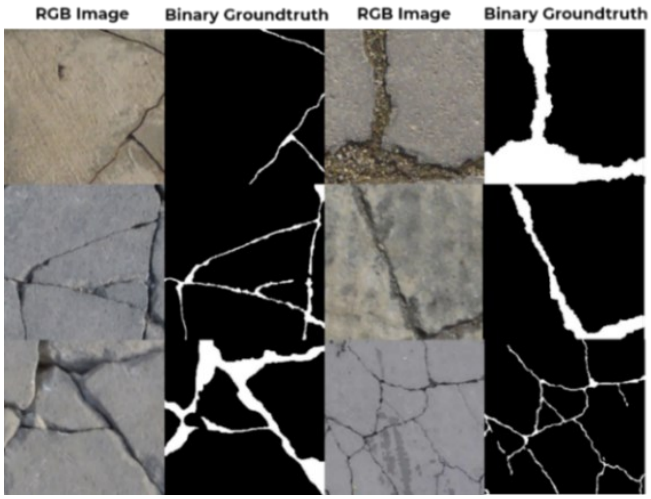

Figure 9. Training dataset.

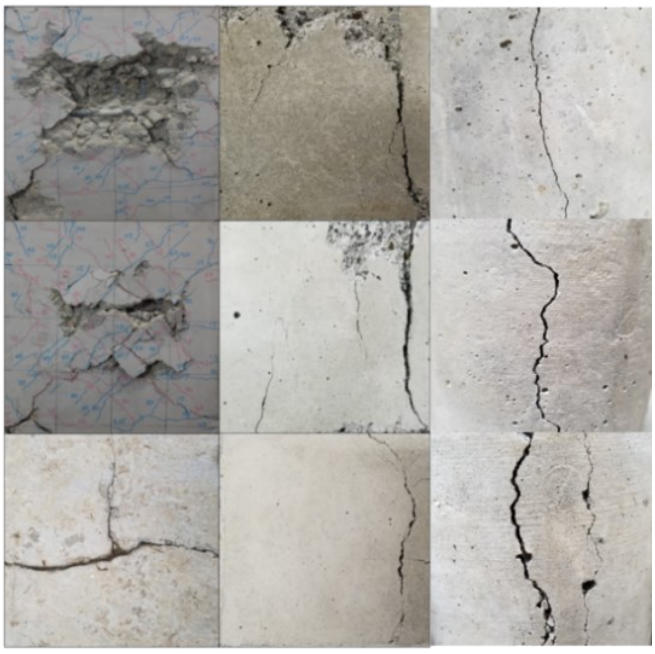
Figure 10. External data.

2.11 Hardware and software configuration

In addition, U-Net and DeepLabV3+ were reconstructed using the Tensorflow framework in the adopted custom-configured Google colab system, supported by the NVIDIA Tesla V100 GPU and 27.4 GB of random-access memory. These configurations are capable of massive parallel computing, which is needed for training large models. The software configuration is shown in Table 1 with models trained in end-to-end fashion without pre-trained backbones. All convolutional weights are set with He normal (He *et al.*, 2015) used as a kernel initializer to further improve the initialization of a non-linear neural network.

Table 1. Software configuration

Software	Version
Keras	2.4.3
Tensorflow	2.3.0
Python	3.6.7
Numpy	1.18.5

3 RESULTS

Dotted and solid lines in Figures 11 to 12 represent validation and training curves, respectively. Green and blue lines represent DeepLabV3+ and U-Net, respectively. All models were trained using 50 epochs. Model weights with the highest F1 score across epochs were selected as a checkpoint. This simply means that every weight was saved on its corresponding epoch. Figure 11 shows the loss curves of U-Net and DeepLabV3+ over epochs. Conversely, the training metrics improvement is shown in figure 12. Training and validation results are shown in Tables 2 and 3 accordingly. The U-Net training results for accuracy, F1, precision, and recall values are 96.57%, 87.55%, 88.15%, and 88.94%. Meanwhile, the DeepLabV3+ training results for accuracy, F1, precision, and recall values are 96.47%, 85.29%, 92.07%, and 81.84%, respectively. The U-Net validation results for accuracy, F1, precision, and recall values are 96.45%, 83.51%, 84.37%, and 84.57% respectively. The DeepLabV3+ validation results for accuracy, F1, precision, and recall values are

95.95%, 79.37%, 81.30%, and 81.60% respectively. Each result shows an acceptable balance between precision and recall metrics. Therefore, the benchmarking decision is inferred from the F1 metric. The high accuracy value of each result shows a dominant correct prediction for the background, which is reasonable from a highly unbalanced segmentation context. Considering the F1 metric, both training and validation results indicate that U-Net is slightly better than DeepLabV3+. The F1 score of U-Net is 2.26% higher than DeepLabV3+ in the training set. Additionally, the F1 score of U-Net is 4.14% higher than DeepLabV3+ in the validation set.

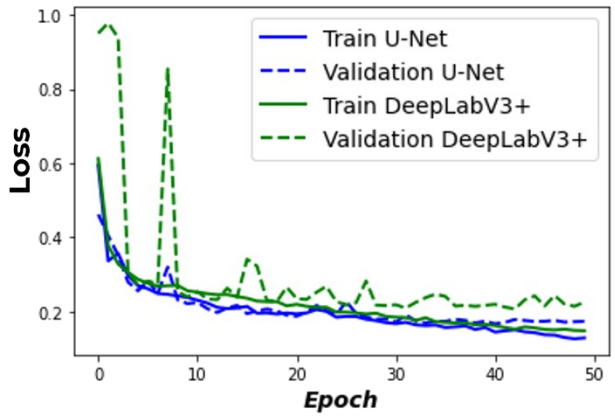

Figure 11. Loss curve.

Figure 11 shows a declining loss trend in a 2D manner, contrary to the F1 curve stated in Equation (13). As shown in Figure 12, U-Net converges faster in accordance with a smoother trend than DeepLabV3+. U-Net achieves 80% F1 by the 12th epochs, which is relatively early because no transfer learning was utilized. One basic key to evaluating the convergence of neural networks is to compare the training and validation trends as defined by solid and dotted lines accordingly. U-Net does not show an overfitting problem as determined by the small margin between the validation and the training F1 curves. A slight overfitting possibility was noticed in the DeepLabV3+ trend. The DeepLabV3+ F1 curve shows the potentials of overfitting from the 20th epoch as the training curve shows a small, steady divergence from that of the validation. This is explained by examining the precision and recall curves. The recall curve does not show any sign of overfitting. However, as shown in the precision curve, the training curve of DeepLabV3+ shows a minor overfitting tendency, which is cross-examined based on the precision result in Tables 2 and 3. A training and validation precision score of 92.07% and 81.30% contributes to the overfitting tendency of the training F1 score in DeepLabV3+. Therefore, U-Net is better when compared to DeepLabV3+ in terms of overfitting tendency.

Table 2. Training Results Evaluation

Metrics	U-Net	DeepLabV3+
Accuracy	96.57%	96.47%
F1	**87.55%**	85.29%
Precision	88.15%	92.07%
Recall	88.94%	81.84%

Table 3. Validation Results Evaluation

Metrics	U-Net	DeepLabV3+
Accuracy	96.45%	95.95%
F1	**83.51%**	79.37%
Precision	84.37%	81.30%
Recall	84.57%	81.60%

Compared to FCN, which Yang *et al.* (2018) tested, both U-Net and DeepLabV3+ are superior for crack segmentation. FCN, U-Net, and DeepLabV3+ achieved training F1 scores of 79.95%, 87.55%, and 85.29%, respectively. Clearly, U-Net and DeepLabV3+ outperform FCN in crack segmentation task.

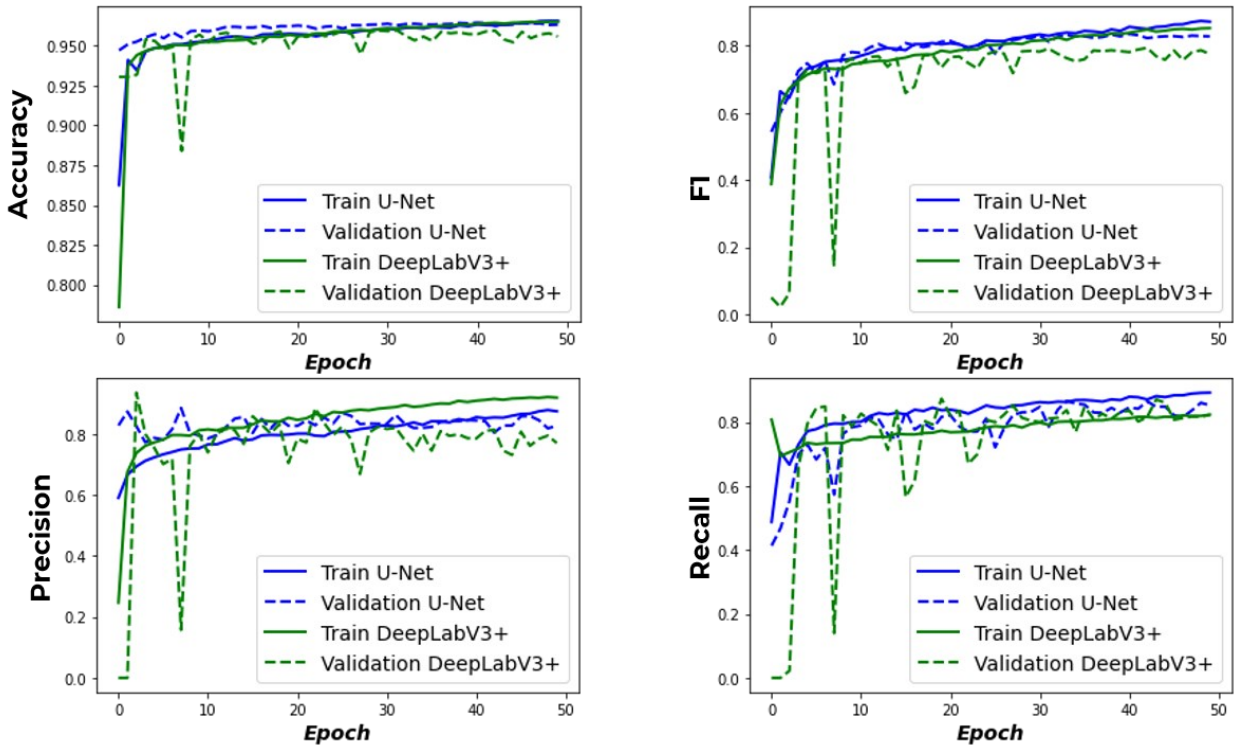

Figure 12. Accuracy, F1, precision, and recall curves.

4 DISCUSSION

Model performance was investigated using internal and external data to study the generalization level that neural networks tend to perform. Internal data is comprised of images with similar distribution as the training data. On the contrary, external data is comprised of images with relatively complex scenarios, which are conceivably distinctive from the training data. The neural network's prediction for internal data is shown in Figure 13, while neural network's prediction for external data is shown in Figure 14. Both U-Net and DeepLabV3+ perform relatively better on internal data. This is expected based on the neural network's tendency to generalize on the fitting data. U-Net is able to segment images with a higher level of granularity than DeepLabV3+. This simply means that DeepLabV3+ tends to yield coarser-grained segmentation edges. Furthermore, segmentation results from DeepLabV3+ are smoother and thicker. A detailed granularity comparison is shown in Figure 15.

Neural networks tend segment complex cracks on external data with certain limitations. As shown in Figure 16, some surface voids were segmented as cracks by both models. These obstacles could be resolved by creating a more diverse training databank consisting of various surface damages. A series of data augmentation can be configured using various transformations for an overall better model fitting.

U-Net is also able to segment thinner and smaller cracks than DeepLabV3+ on both internal and external data, as shown in Figure 17. This is arguably due to the concatenating features of every level in U-Net to recover fine-grained details of thin objects like cracks, which aids in obtaining more detailed information. However, extremely thin cracks ranging from 1 to 3 pixels wide with slightly blurred images are often undetectable on both architectures, as shown in Figure 18. This is potentially due to the loss of spatial information in the deeper feature maps caused by max-pooling and striding operations. However, more studies are required to validate these hypotheses.

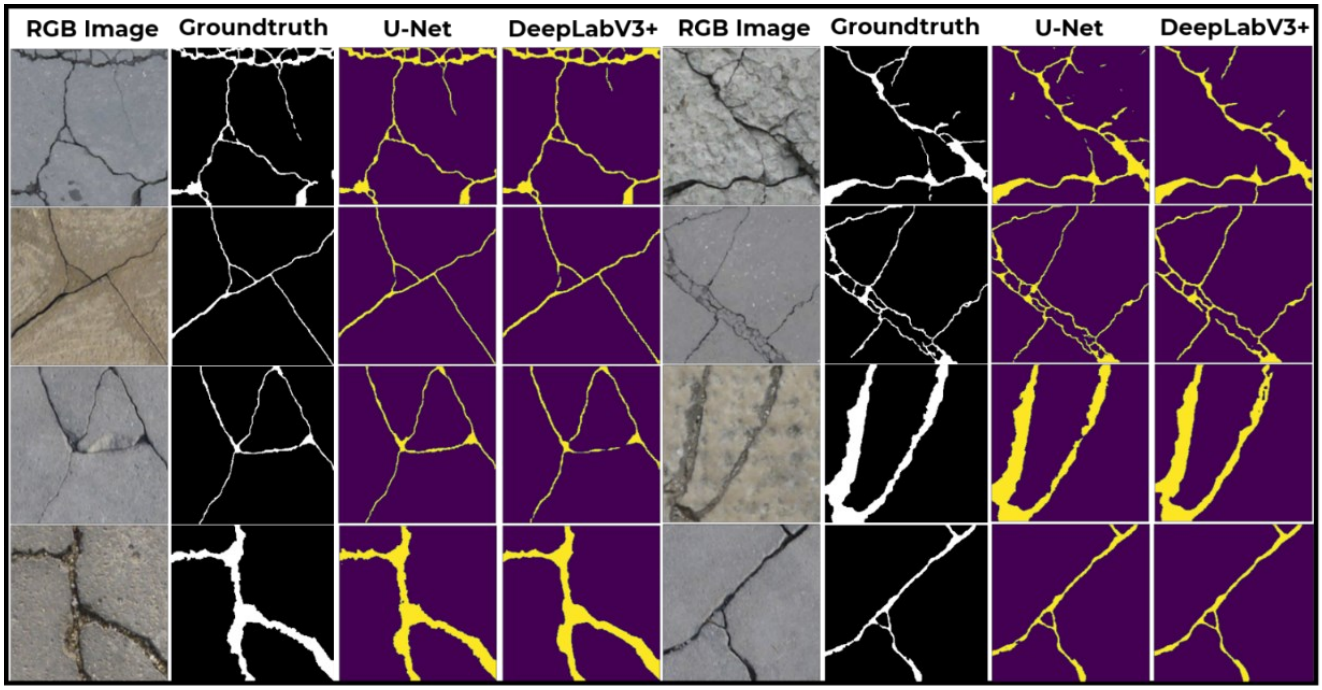

Figure 13. Segmentation results comparison on internal data.

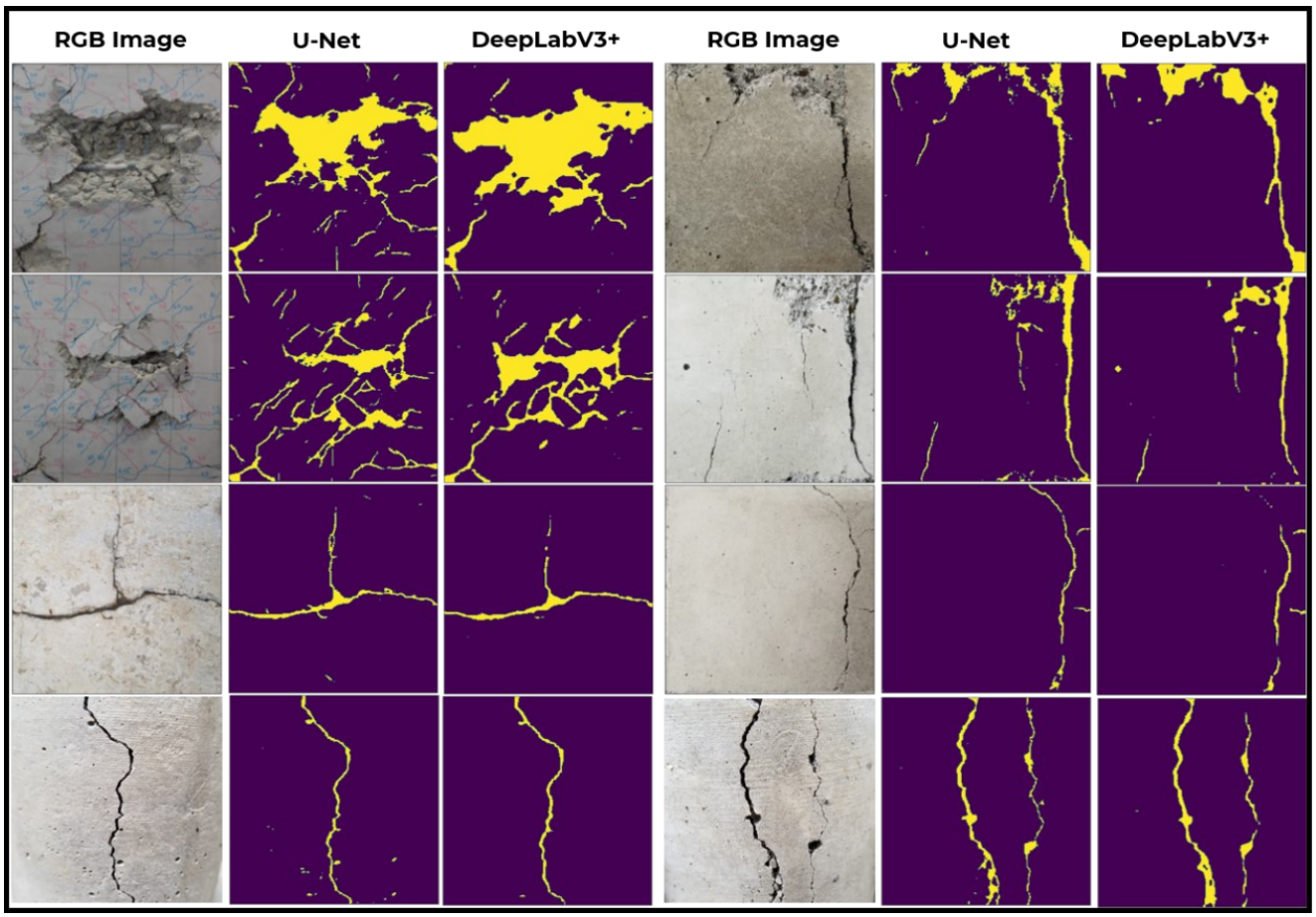

Figure 14. Segmentation results comparison on external data.

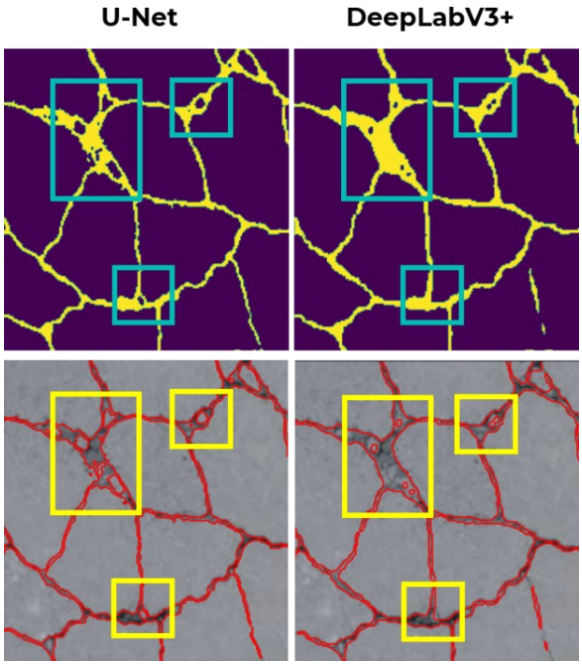

Figure 15. Granularity comparison.

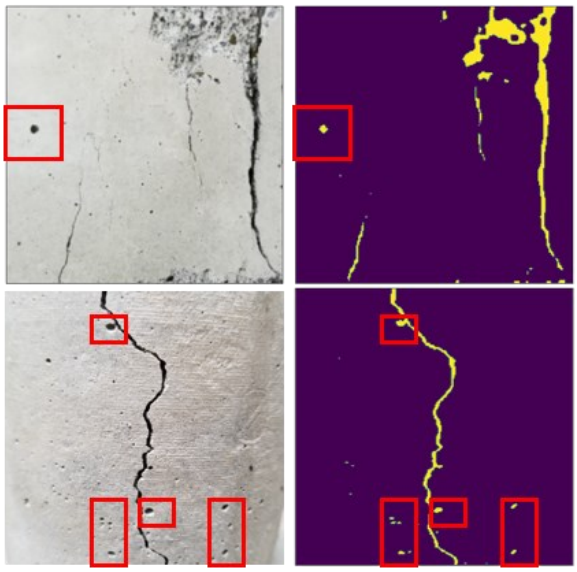

Figure 16. Surface voids detected as cracks.

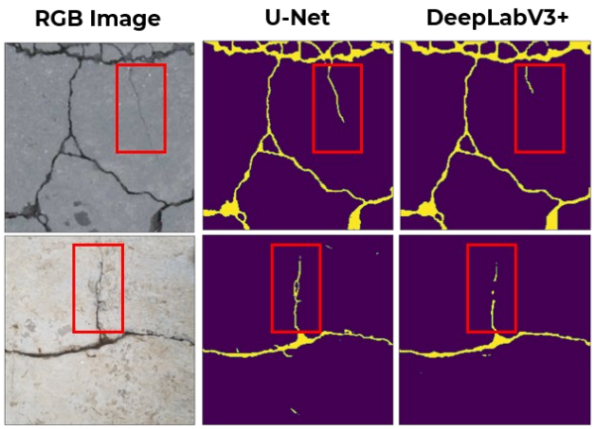

Figure 17. Thin crack segmentation results comparison.

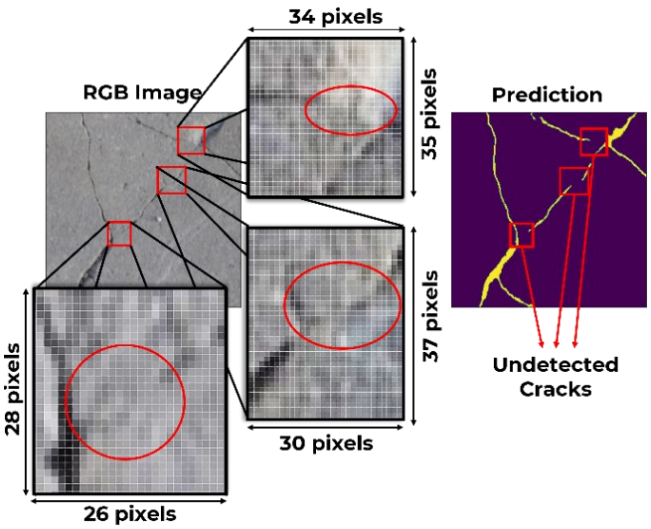

Figure 18. Undetected cracks.

5 CONCLUSION

In this study, 2 novel encoder-decoder CNN architectures, namely U-Net and DeepLabV3+ were implemented to detect cracks on concrete surfaces. U-Net and DeepLabV3+ achieved a fast and high-performing convergence with a training F1 score of 87.55% and 85.29% respectively. Checkpoint results prove the advantages of U-Net and DeepLabV3+ over FCN. Test results suggest both models performed relatively better on internal data due to the common generalization tendency. Neural networks are also able to segment cracks on external data with excellent results. This indicates the ability of both architectures to segment cracks in various environments. However, with the consideration of surface voids on external data, neural networks often irregularly categorize these as cracks. Therefore, a richer and larger dataset can be constructed to improve a trained crack detector significantly. Additionally, U-Net was able to segment thinner cracks than DeepLabV3+ on both internal and external data, correlated to the finer-grained segmentation result from U-Net.

DISCLAIMER

The authors declare no conflict of interest.

AVAILABILITY OF DATA AND MATERIALS

All data are available from the author.

ACKNOWLEDGMENTS

The authors are grateful to the Department of Civil Engineering, Universitas Katolik Parahyangan, for supporting this research.

REFERENCES

Chen, L. C., Papandreou, G., Schroff, F., & Adam, H., 2017. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv:1706.05587*.

Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H., 2018. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 801-818.

Chollet, F., 2017. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251-1258.

Goutte, C., & Gaussier, E., 2005. A Probabilistic Interpretation of Precision, Recall And F-Score, With Implication for Evaluation. In *European conference on information retrieval*, pp. 345-359. Springer, Berlin, Heidelberg.

He, K., Zhang, X., Ren, S., & Sun, J., 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026-1034.

Ioffe, S., & Szegedy, C., 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International conference on machine learning*, pp. 448-456. PMLR.

Kingma, D. P., & Ba, J., 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Long, J., Shelhamer, E., & Darrell, T., 2015. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431-3440.

Milletari, F., Navab, N., & Ahmadi, S. A., 2016. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pp. 565-571. IEEE.

Nair, V., & Hinton, G. E., 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Icml*.

Qian, N., 1999. On the Momentum Term in Gradient Descent Learning Algorithms. *Neural networks*, *12*(1), pp. 145-151.

Ronneberger, O., Fischer, P., & Brox, T., 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234-241. Springer, Cham.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R., 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The journal of machine learning research*, *15*(1), pp. 1929-1958.

Yang, X., Li, H., Yu, Y., Luo, X., Huang, T., & Yang, X., 2018. Automatic Pixel-Level Crack Detection and Measurement Using Fully Convolutional Network. *Computer-Aided Civil and Infrastructure Engineering*, *33*(12), pp. 1090-1109.