Research Article

Science, Engineering and Health Studies
https://li01.tci-thaijo.org/index.php/sehs
ISSN (Online): 2630-0087

Comparison of DeepLab v3+ base networks for crack segmentation under limited computational resources

Thitiporn Lertrusdachakul[1] and Pierre-Emmanuel Leni[2]*

[1] Multimedia Technology Program, Faculty of Information Technology, Thai-Nichi Institute of Technology, Bangkok 10250, Thailand
[2] Chrono-Environment Laboratory, University of Franche-Comté, Montbéliard 25200, France

***Corresponding author**:
Thitiporn Lertrusdachakul
thitiporn@tni.ac.th

***Received**: 20 June 2024*
***Revised**: 27 January 2025*
***Accepted**: 22 February 2025*
***Published**: 30 December 2025*

***Citation**:*
Lertrusdachakul, T., & Leni, P.-E. (2025). Comparison of DeepLab v3+ base networks for crack segmentation under limited computational resources. Science, Engineering and Health Studies, 19, 25020009.

ABSTRACT

Accurate crack segmentation plays a crucial role in infrastructure assessment and preventive maintenance. This research explored the crack segmentation efficacy of DeepLab v3+, a modern and advanced semantic segmentation network with a high performance and reduced computational cost. The performance comparison was investigated of DeepLab v3+ with different base networks, including Inception-ResNet-v2, Xception, ResNet-50, MobileNet-v2, and ResNet-18. The objective of this paper was to recommend the base network and its optimizer of DeepLab v3+ architecture in terms of crack segmentation of structure for structural health assessment and monitoring under limited resources. The optimizer algorithm, mini-batch size, learning rate, and squared gradient decay factor were adjusted to obtain the best model for each base network considering limited resources of graphics processing unit (GPU) for model training. The best results were analyzed in terms of mean accuracy, class accuracy, and weighted IoU whilst taking the model size into account. The recommended models ranked from the most accurate to the smallest in size are DeepLab v3+ network based on ResNet-50 with Adam optimizer, Xception with RMSProp optimizer, ResNet-18 with SGDM optimizer, and MobileNet-v2 with RMSProp optimizer, respectively. The findings assist in choosing a suitable network architecture for specific applications considering the compromise between model size and performance. The results also highlight the feasibility of the network architecture with tested conditions in terms of structural crack segmentation under limited computational resources.

Keywords: crack segmentation; DeepLab v3+; deep learning; optimization algorithm

1. INTRODUCTION

Cracks on the surface of concrete structures act as early indicators of structural deterioration, emphasizing the need for timely maintenance to prevent severe damage to the concrete structure. Crack segmentation of concrete surfaces is a challenging task in civil engineering due to the complex and varied appearance of cracks. It can be used to identify and assess the severity of cracks in concrete structures. Cracks can lead to structural failure, so it is important to be able to detect them early and accurately. There are a number of different methods that can be used for crack segmentation, including deep learning, machine learning and conventional image processing approaches. Conventional image processing approaches, such as thresholding (Talab et al., 2016) and edge detection, can be effective for simple cracks, but they can struggle with more complex cracks, such as those with low intensity or significant noise characteristics (Mohan & Poobal, 2018; Kheradmandi & Mehranfar, 2022).

In recent years, deep learning has emerged as a promising new approach for crack segmentation (Li et al., 2022; Xu et al., 2023; Yang et al., 2023; Mei & Gül, 2020). These techniques are able to learn from data, which means that they can improve their accuracy over time. The techniques of deep learning have proven to be particularly effective in crack segmentation because they are capable of learning complex patterns from images.

One of the most common deep learning architectures for crack segmentation is the convolutional neural network (CNN). CNN is a kind of neural network that works well for image analysis applications. It is capable of extracting characteristics and features from images, such as shapes, textures, and edges. These features can be then used to classify pixels as either cracks or non-cracks. Several studies have investigated the use of CNN for crack segmentation of concrete surfaces, roads and pavements. These studies have shown that CNN can achieve high accuracy in crack segmentation, even for images with complex or cluttered backgrounds (Liu et al., 2019; Su & Wang, 2020; Liu et al., 2020; Nguyen et al., 2021; Kim et al., 2021; Han et al., 2022).

Other deep learning architectures have also been used for crack segmentation. For example, a kind of CNN called U-Net architecture is specifically designed for image segmentation applications. The U-Net and U-Net-based architectures have exhibited remarkable proficiency in crack segmentation, as it is able to learn complex patterns from images and segment cracks accurately and cope with noise and variations in crack appearance robustly (Lau et al., 2020; Cui et al., 2022; Su et al., 2022).

Another promising deep learning model for crack segmentation of concrete surface is DeepLab v3+ model, which is the advanced model for semantic segmentation that can be used to classify pixels in an image into different categories. It has demonstrated superior performance on large-scale crack segmentation datasets, especially in scenarios with diverse crack patterns and complex backgrounds (Fu et al., 2021; Pu et al., 2022; Sun et al., 2022; Zhou et al., 2023). The DeepLab v3+ has strength in capturing multi-scale contextual information and handling large-scale datasets. It utilizes atrous separable convolution, which reduces computational cost while maintaining feature extraction capabilities. Therefore, DeepLab v3+ was selected for further study which is a recent advanced deep learning architecture that has not been studied on various base network families in crack segmentation. Based on the model size, accuracy, and previous researches on crack segmentation related to the base network of DeepLabv3+ (Nguyen et al., 2024; Xie et al., 2024), five competitive base networks were finally selected for the comparison, i.e., Inception-ResNet-v2, Xception, ResNet-50, MobileNet-v2, and Res-Net-18. The training options are optimized under limited resource conditions for the best model of crack segmentation. The model results and the associated analysis will be presented as a guideline for applying in structural crack segmentation and model development of new structures under limited resources.

2. MATERIALS AND METHODS

2.1 Related deep learning models

The core deep learning model used in this research is DeepLab v3+ with five base convolutional neural networks, i.e., Inception-ResNet-v2, Xception, ResNet-50, MobileNet-v2, and Res-Net-18, which are summarized in this section.

2.1.1 DeepLab v3+

DeepLab v3+ is a CNN-based architecture for image segmentation. DeepLab v3+ uses a combination of several cutting-edge techniques to achieve outstanding performance on image segmentation tasks. These techniques include (Chen et al., 2018):

Atrous spatial pyramid pooling (ASPP) module: Accurate segmentation of objects with varying sizes and shapes is made possible by the ASPP module, which enables the network to learn features from multiple scales.

Encoder-decoder structure: In the model, the decoder part generates the segmentation mask, while the encoder part extracts the features from the input image. The use of an encoder-decoder architecture allows DeepLab v3+ to learn more complex relationships between pixels in the image and capture both global and local context, which is essential to accurately segment the complicated scenes.

Atrous convolution: With atrous convolution, the network can learn features and characteristics from large spatial contexts without impacting computational cost.

Figure 1 shows the architecture of DeepLab v3+ with base network. In the encoder part, the image is inputted to the base network and two different layers are extracted. One is connected to ASPP and another one goes to the decoder part. The ASPP applied three different dilation rates (atrous convolution with an atrous rate of 6, 12, 18, respectively) for capturing multi-scale information. The output is then concatenated and fed through a 1×1 convolution before going to the upsampling by a factor of 4 in the decoder part. The output of upsampling is concatenated with the features from the encoder part that pass through a 1×1 convolution. The output from concatenation is fed through a 3×3 convolution before again upsampling by a factor of 4 in the last stage of the decoder part.

The ASPP in the encoder part of DeepLab v3+ capture rich contextual information from the base network by feature pooling at different resolutions. The decoder part up-samples the deep feature maps to enhance the spatial resolution and fuses them with shallow features from the base network of the encoder part to optimize the precise position of the crack segmentation. This encoder-decoder architecture effectively refines segmentation boundaries and contributes to improve overall segmentation performance.

The feature extraction in DeepLab v3+ is mainly done on the base network (sometimes called backbone network) which will be briefly described in section 2.1.2 to section 2.1.6 for the five selected base networks of this research.

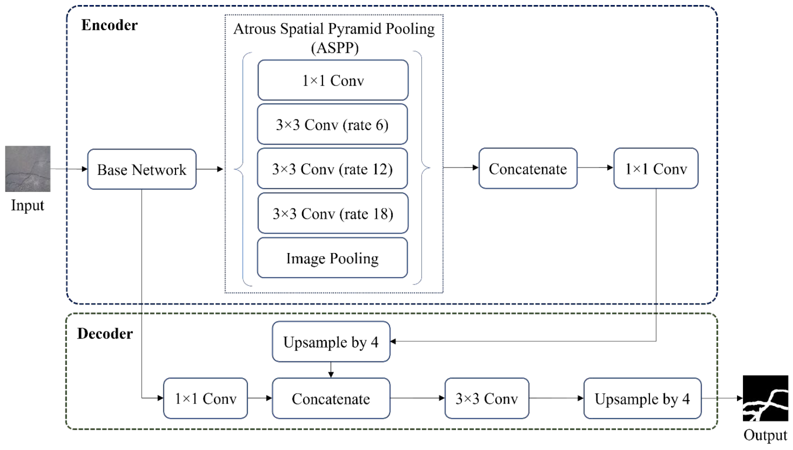


Figure 1. Architecture of DeepLabv3+ with base network

2.1.2 ResNet-18

ResNet-18 (He et al., 2016) is a CNN that is 18 layers deep based on the idea of residual connections. Residual connections are also known as "skip connections" or "shortcut connections". They are a way of connecting the output of one layer of the network to the input of another layer, even if there are several layers in between. Residual connections allow the network to learn very deep representations or more complex features without a vanishing gradient issue, which can occur in traditional CNNs. This makes ResNet-18 a very powerful and accurate model for image recognition tasks.

ResNet-18 is comprised of a number of convolutional blocks, each block consisting of two or three convolutional layers, a batch normalization layer and a ReLU activation function. The convolutional blocks are connected to each other using skip connections, which allow the network to learn long-range dependencies in the input data. ResNet-18 is a relatively small model, with only around 11 million parameters. This makes it easy to train and deploy on devices with limited computational resources. The layer that connects to the ASPP in the encoder part of DeepLab v3+ is "res5b_relu" (28×28×512). The shallow features that are fed to the decoder part of DeepLab v3+ in Figure 1 are from layer "res2b_relu" (112×112×64).

2.1.3 ResNet-50

ResNet-50 (He et al., 2016) is a 50-layer deep CNN that is based on the use of residual connections. As many more layers are added to the network for ResNet-50, bottleneck blocks are used. The utilization of these blocks facilitates a reduction in network parameters, thereby enhancing computational efficiency during the training process.

ResNet-50 is a popular choice for image recognition tasks because it is both powerful and efficient. It has around 25.6 million parameters, which is more than ResNet-18, but still relatively small compared to other deep learning models. This makes it easier to train and deploy on GPUs and CPUs. ResNet-50 generally has better accuracy than ResNet-18, but it is also larger, slower and requires more memory to train. ResNet-18 is a good choice for mobile devices or resource-constrained applications, while ResNet-50 is a better choice for high-performance tasks where accuracy is the top priority. The features from layer "activation_49_relu" (28×28×2048) are connected to the ASPP in the encoder part of DeepLab v3+ and the layer "activation_10_relu" (112×112×256) is fed to the decoder.

2.1.4 MobileNet-v2

MobileNet-v2 (Sandler et al., 2018) is a lightweight CNN with 53 layers deep (default configuration). It is significantly smaller and faster than other CNNs, making it suitable for mobile and embedded applications. MobileNet-v2 improves upon the original MobileNet architecture by introducing several key innovations, including the following:

Inverted residual blocks: These blocks utilize a "bottleneck" design that reduces the number of channels in the intermediate layers, leading to significant computational efficiency.

Linear bottlenecks: The bottlenecks in MobileNet-v2 use a linear activation function, such as ReLU, instead of a non-linear activation function like sigmoid or tanh. This further reduces the computational cost of the network. The width of the bottleneck layers is determined by a "width multiplier" parameter, allowing the model to be scaled for different accuracy and resource constraints.

Depthwise separable convolutions: This technique factorizes the standard convolution operation into two distinct steps: a depthwise convolution for feature extraction within each input channel and a pointwise convolution for channel-wise combination. This factorization leads to a significant reduction in both computational complexity and network parameters, ultimately improving model efficiency. The layer "block_16_project_BN" (28×28×320) is connected to the ASPP in the encoder part and the layer "block_3_expand_relu" (112×112×144) is fed to the decoder of DeepLabv3+.

2.1.5 Xception

Xception (Chollet, 2017) is a deep CNN architecture that is 71 layers deep. It stands for "extreme inception", aiming to increase the Inception-v3 model's performance by utilizing depthwise separable convolutions and residual connections. The architecture consists of a series of depthwise separable convolutions, which are a more efficient type of convolution than standard convolutions. This reduces the computational cost of the network while maintaining accuracy, allowing for deeper and more efficient networks. Depthwise separable convolutions make Xception more efficient than other deep CNNs, making it suitable for real-world applications with resource constraints.

The Xception architecture also uses residual connections, which are a type of skip connection that helps to address the vanishing gradient problem. In deep neural networks, training can be hindered by the vanishing gradient problem. This phenomenon arises when gradients of the loss function with respect to the weights in earlier layers diminish significantly. Consequently, the network struggles to learn meaningful features in these initial layers. This attenuation impedes the network's ability to learn meaningful features from the input data. Residual connections mitigate the vanishing gradient problem by establishing direct gradient flow between input and output of the layer. This bypass mechanism ensures that gradients propagate more effectively through the network, facilitating the training of deeper architectures and enabling the network to effectively learn complex representations for improved prediction accuracy. The Xception architecture is a popular choice for researchers and developers due to its simplicity, efficiency, and accuracy. The features from layer "block14_sepconv2_act" (28×28×2048) were fed to the ASPP in the encoder part and the layer "add_1" (112×112×128) was fed to the decoder of DeepLab v3+.

2.1.6 Inception-ResNet-v2

Inception-ResNet-v2 is a CNN architecture that takes advantage of two powerful architectures: Inception and ResNet (Szegedy et al., 2017). It combines these two approaches by replacing the filter concatenation stage of the inception architecture with residual connections. The residual connections help to improve the flow of information throughout the network, potentially leading to enhanced performance. The network is enabled to extract more robust feature representations and achieve higher accuracy.

Inception-ResNet-v2 is a 164-layer deep network that uses a combination of convolutional layers, pooling layers, and residual connections. The network comprises several "inception modules", each containing multiple convolution layers with different filter sizes. It uses batch normalization and ReLU activation functions to stabilize training and improve performance. The input image is first processed by several initial convolution layers. Then, the image flows through multiple inception modules, each extracting features at different scales and combining them effectively. Within each inception module, residual connections bypass some layers, allowing information to flow directly from earlier stages to later stages. This process continues through multiple inception modules and residual connections, progressively building more complex representations of the image. Finally, the extracted features are put into a series of fully-connected layers for classification or other tasks. Overall, Inception-ResNet-v2 is a powerful and efficient CNN architecture with a wide range of applications. The features from layer "conv_7b_ac" (28×28×1536) is fed to the ASPP in the encoder part and the layer "activation_5" (112×112×192) is fed to the decoder part of DeepLab v3+.

2.2 Optimization algorithms

Stochastic gradient descent with momentum (SGDM), root mean square propagation (RMSProp), and adaptive moment estimation (Adam) are all optimization algorithms commonly used in machine learning and will be tested for training neural networks in the experiment.

The stochastic gradient descent (SGD) algorithm is a widely used method for finding the minimum of a function. SGD is a simple and efficient optimization algorithm that iteratively refines the parameters of a function by following the negative gradient direction. The gradient is a vector that points in the direction of the steepest descent of the function. However, SGD can be sensitive to the choice of learning rate and can get stuck in local optima. Therefore, three variants of the SGD algorithm (Du, 2019), which are commonly used in neural network training, will be introduced and used in the experiment.

2.2.1 SGDM

SGDM is an extension of SGD that incorporates momentum to accelerate the learning process. Momentum is a decaying velocity term that helps the algorithm move in the right direction and avoid getting stuck in local minima. In the training process of the experiment, the training option for optimizer algorithm is set to be "sgdm" with adjustments to the mini-batch size and learning rate for the best model of image segmentation.

2.2.2 Root mean square propagation (RMSProp)

RMSProp is another extension of SGD that addresses the issue of oscillating gradients. Oscillating gradients can cause the algorithm to zigzag around the optimal solution, making it difficult to converge. RMSProp maintains an estimate of the moving average of the squared gradients for each parameter. This estimate is used to scale the gradients, which helps to prevent them from exploding or vanishing. It helps to smooth out the gradients and improve convergence. RMSProp is often considered to be a more stable and robust algorithm than SGDM. In this case, the training option for optimizer algorithm is set to be "rmsprop" with adjustments to the mini-batch size, learning rate, and squared gradient decay factor. The squared gradient decay factor is the decay rate of squared gradient moving average for RMSProp and we use the typical values of the decay rate of 0.9, 0.99, and 0.999 in the experiment.

2.2.3 Adam

Adam is a popular optimization algorithm that combines the advantages of SGDM and RMSProp. It maintains an exponential moving average of both the squared gradient and the gradient, and it also includes a bias correction term to ensure that the initial estimates of the averages are not too small, which helps to improve the convergence of the algorithm. In general, Adam is the most powerful of the three algorithms, but it is also the most complex. Adam is less sensitive to hyperparameters than SGDM and RMSProp, and it converges faster than SGDM. For this case, the optimizer algorithm in the training process is set to be "adam" with adjustments to the mini-batch size, learning rate, and squared gradient decay factor. We used the same typical decay rate to the RMSProp, i.e., 0.9, 0.99, and 0.999 in the experiment.

Ultimately, the best algorithm for any specific circumstance will rely on particular characteristics of the circumstance and the desired trade-offs between convergence speed, stability, and hyperparameter tuning.

2.3 Methodology

DeepLab v3+ demonstrates an effective ability to capture contextual information at multiple scales, which significantly benefits its performance on large-scale datasets. It exhibits commendable computational efficiency. This efficiency enables training and utilization on graphics processing units (GPUs) with limited memory resources, further expanding its application potential. These characteristics, combined with its proven success in image segmentation tasks, motivated our focused study on DeepLab v3+ for transfer learning of crack segmentation. The objective of this research is to compare its base networks of five widely and potentially used CNN-based deep neural networks and find the best training parameters under resource limitation. This is to facilitate its application and practical implementation of the new model for diverse structures in resource-constrained environments. Although a high-performance GPU is desirable for training the neural network model, it requires great expense, making it difficult for the general public or small labs and organizations to participate in this area of study. Therefore, this work focusses on working under limited hardware resources for model training, i.e., the number, memory size, and performance of the GPU. The hardware used in this research for training the neural network is a single GPU: NVIDIA® GeForce RTX™ 3070 with 8GB of GPU memory.

In this study, we experiment on DeepLab v3+ with base CNNs of Inception-ResNet-v2, Xception, ResNet-50, MobileNet-v2, and Res-Net-18. The architecture of DeepLab v3+ with a base network is fundamentally based on Figure 1. The crack segmentation process consists of model training, finding the best parameters, and model evaluation as illustrated in Figure 2.

2.3.1 Model training

In this step, a Kaggle public dataset was utilized (https://www.kaggle.com/datasets/lakshaymiddha/crack-segmentation-dataset), which was merged from 12 available crack segmentation datasets. It contains 11,298 images of 448 × 448 pixels, which includes images with crack and no crack pixel and their masks. The mask image is binary: the crack pixel with white color on black background. The dataset was divided into 20% test dataset, 20% validation dataset, and 60% training dataset. The data splitting is stratified for minimizing bias due to imbalanced distributions from 12 crack segmentation datasets. This was to ensure that the test, training, and validation sets contain similar proportions of each dataset.

The use of data augmentation technique involves randomly modifying the training dataset to increase network accuracy. The same random for X/Y translation of +/- 10 pixels and left/right reflection was applied to the original data for both crack image and pixel label data (mask image) during the training for data augmentation.

Since the classes (crack and no crack) are not balanced because crack pixels appear with much less area in the image, this imbalance can adversely affect the learning process because the learning is biased in favor of the dominant class. To handle this issue and improve the training, class weighting obtained from dataset statistics was used to balance the classes. Each class's allocated weight was calculated based on median frequency, as shown in Equation 1. The obtained class weight is applied to the last layer of the neural network, i.e., pixel classification layer.

$$classWeight(c) = \frac{median(imageFreq)}{imageFreq(c)} \qquad (1)$$

where $imageFreq(c)$ indicates the division of the number of pixels in the class by the total number of pixels in images containing an instance of the class (c).

Five distinct CNN models were utilized for the base network training of DeepLab v3+, i.e., Inception-ResNet-v2, Xception, ResNet-50, MobileNet-v2, and Res-Net-18. The DeepLab v3+ of these base networks was trained with weights initialized from a pretrained base network. These base networks are trained on more than a million images from the ImageNet database (http://www.image-net.org). The validation patience was set to 4. This value indicates the number of times that the loss on the validation set can be more than or equal to the lowest loss that occurred previously before neural network training ends. This is to prevent the network from overfitting on the training dataset by stopping the training early when the validation accuracy converges.

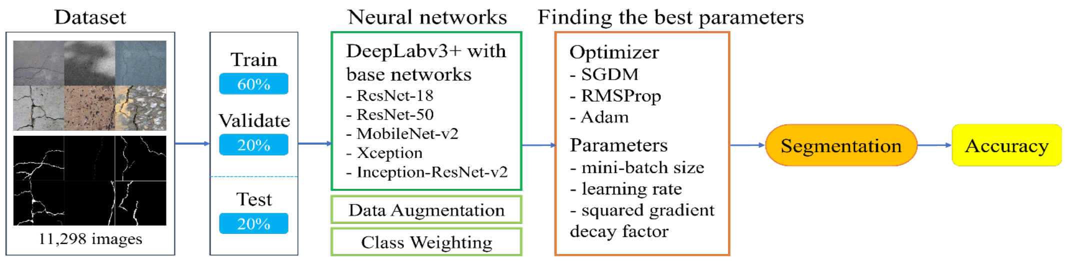


Figure 2. Crack segmentation process of DeepLab v3+ with five base CNN models

2.3.2 Finding the best parameters

We compared three optimization algorithms for training networks: SGDM, RMSProp and Adam. From the preliminary experiment, three parameters were adjusted for the training. The first parameter was mini-batch size which determines how big of a mini-batch to employ in each training iteration. A mini-batch is a subset of the training set that is used to evaluate the gradient of the loss function and update the weights. Next, the best result was adjusted with the second parameter of initial learning rate. For RMSProp and Adam optimizers, the squared gradient decay factor was varied, which is the decay rate of squared gradient moving average for RMSProp and Adam optimizers. The value is specified as a nonnegative scalar less than 1. The typical values of 0.9, 0.99, and 0.999 were tested, which are corresponding to averaging lengths of 10, 100, and 1000 parameter updates, respectively. This was to find the best training parameters to obtain the best result for each model.

2.3.3 Model evaluation

The performance of the trained model was evaluated on a test dataset and can be measured by using data set and class metrics. For the application of crack segmentation, the mean accuracy and weight IoU (Intersection over Union) from data set metrics and the accuracy of each class from class metrics were used to assess the model performance.

The accuracy is the proportion of each class's correctly classified pixels to the total number of pixels in that class, determined by the ground truth, which can be expressed as Equation 2.

$$Accuracy = \frac{TN+TP}{FN+FP+TN+TP} \tag{2}$$

The terms are defined as shown in Figure 3, where true negative (TN) is the number of actual negative samples that were correctly predicted as negative; true positive (TP) is the number of actual positive samples that were correctly predicted as positive; false negative (FN) is the number of actual positive samples that were incorrectly predicted as negative; and false positive (FP) is the number of actual negative samples that were incorrectly predicted as positive. The mean accuracy is the average accuracy of all classes across all images.

The weighted IoU was determined by the average IoU of all classes, weighted by the number of pixels in the class, where the IoU of each class was the proportion of correctly classified pixels to all predicted and ground truth pixels in that class, which can be expressed as Equation 3.

$$IoU = \frac{TP}{FN+FP+TP} \tag{3}$$

The weighted IoU was used to reduce the impact of errors in the small classes on the aggregate quality score when the images had disproportionally sized classes.

Beside the focus on class accuracy, the mean accuracy and the weighted IoU which provide an overview of the network performance, were investigated to determine the best model.

3. RESULTS

In this paper, DeepLab v3+ was trained with five base networks, i.e., Inception-ResNet-v2, Xception, ResNet-50, MobileNet-v2, and Res-Net-18 using three optimizers. The mini-batch size, learning rate and squared gradient decay factor were adjusted during the neural network training to compare the results and determine the best model under limited resource availability.

Figure 4 shows an example result of training accuracy for DeepLab v3+ with a ResNet-18 base network. The best result was obtained using the Adam optimizer with a mini-batch size of 32 and a learning rate of 0.00001. The assigned squared gradient decay factor for Adam optimizer was set to 0.999. The training stopped at 1,450 iterations when the validation criteria were met.

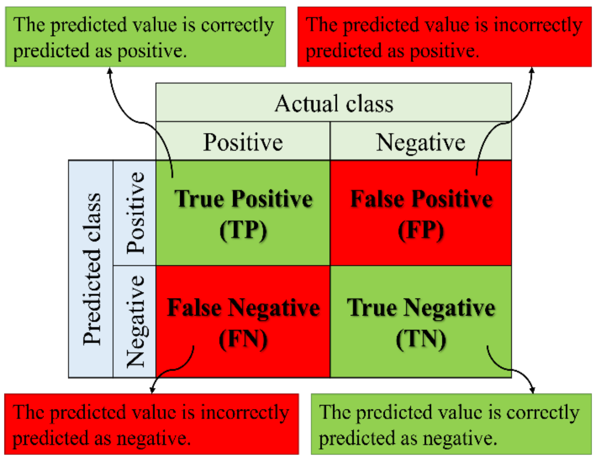

The predicted value is correctly predicted as positive.
The predicted value is incorrectly predicted as positive.
Actual class
Positive
Negative
Predicted class
Positive
Negative
True Positive (TP)
False Positive (FP)
False Negative (FN)
True Negative (TN)
The predicted value is incorrectly predicted as negative.
The predicted value is correctly predicted as negative.

Figure 3. Confusion matrix

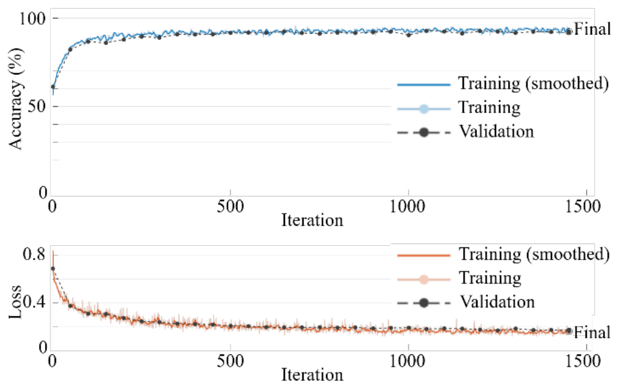


Figure 4. DeepLab v3+ with base network of ResNet-18 measured performance (Adam optimizer)

The best results of accuracy for crack class and background class, the mean accuracy and the weighted IoU are summarized in Table 1 for each base network and all tested optimizers. The training parameters for each model are described in Table 2. In Table 1, the highest values for each base network and all networks are highlighted with bold and underline, respectively. The green highlight shows the maximum value of each optimizer for five base networks. When comparing the best results in each base network, the recommended model is the highest result of mean accuracy or weighted IoU together with the highest result of crack accuracy or background accuracy, i.e., having at least two highest values. Note that all the four values should be at least 0.9 for the acceptable results. Therefore, the recommended models for each base network are indicated in cyan color in Table 1. For base networks of ResNet-18 with SGDM optimizer and Xception with RMSProp optimizer, the results obviously show that they have both highest mean accuracy and highest weighted IoU including highest background accuracy (bold values). For the ResNet-50 base network, both SGDM and Adam optimizers are recommended as they have two highest values out of four. In MobileNet-v2 base network, the model with RMSProp optimizer is recommended because it has the highest weighted IoU and background accuracy. There is no recommended model for Inception-ResNet-v2 base network because the crack accuracy does not meet the criteria, i.e., less than 0.9. Table 3 describes the rank of recommended models. The ResNet-50 base network with Adam optimizer was ranked as the first recommendation because of its highest crack accuracy and mean accuracy from all fifteen models (underline values). The second rank was the Xception base network with RMSProp optimizer as it had the highest background accuracy and weighted IoU in all models. The ResNet-50 base network was also ranked as the third recommendation with SGDM optimizer because it had high values of crack accuracy, background accuracy, mean accuracy, and weighted IoU. All the three recommended models were also the base models that included the highest results in that optimizer testing under limited resources (green-highlight values). The fourth and fifth recommended models were the ResNet-18 base network with SGDM optimizer and the MobileNet-v2 base network with RMSProp optimizer, respectively. These two models had good results with lower sized models. The performance of the best five models of DeepLab v3+ in crack segmentation is summarized in Table 4. It can be seen that all the models satisfy having at least two highest values in each base network (bold values), i.e., one in class accuracy (crack accuracy or background accuracy) and another one in mean accuracy or weighted IoU. The values are all greater than 0.9 as the minimum acceptable result. The first rank has the highest mean accuracy and crack accuracy while the second rank had the highest background accuracy and weighted IoU from all base networks (underline values). For the third rank, it did not have any highest values from all base networks (no underline value), but it had high values of mean accuracy, crack accuracy, background accuracy, and weighted IoU, which were all higher than the values of the fourth and the fifth ranks. For the fourth rank, it also had all evaluated values higher than the results of the fifth rank.

Table 1. Summarization of the best performance for DeepLab v3+ with five base networks and three optimizers under resource limitation

No.	Deeplab v3+ with base network	Number of learnable parameters (millions)	Number of layers	Optimizer	Accuracy [crack] [background]	Mean accuracy	Weighted IoU	Model size (MB)
1	ResNet-18	20.6	100	SGDM	0.95001 **0.94230**	**0.94616**	**0.92063**	58.4
2				RMSProp	0.95988 0.93178	0.94583	0.90965	58.5
3				Adam	**0.96160** 0.92299	0.94229	0.90035	58.4
4	ResNet-50	43.9	206	SGDM	0.95502 **0.94417**	0.94960	**0.92291**	141
5				RMSProp	0.95343 0.94079	0.94711	0.91912	141
6				Adam	**0.96944** 0.93151	**0.95048**	0.90977	142
7	MobileNet-v2	6.7	186	SGDM	**0.95089** 0.93374	0.94232	0.91135	9.44
8				RMSProp	0.94354 **0.94031**	0.94192	**0.91814**	9.50
9				Adam	0.94966 0.93782	**0.94374**	0.91571	9.50
10	Xception	27.6	205	SGDM	0.93684 0.93807	0.93746	0.91540	83.4
11				RMSProp	0.94789 **0.94642**	**0.94716**	**0.92507**	84.2
12				Adam	**0.94892** 0.94212	0.94552	0.92037	84.2
13	Inception-ResNet-v2	71.1	853	SGDM	0.90794 0.92972	**0.91883**	0.90515	238
14				RMSProp	**0.91918** 0.89852	0.90885	0.87317	235
15				Adam	0.86940 **0.93177**	0.90059	**0.90564**	240

Note:
Bold: The highest values in each base network.
Underline: The highest values of all base networks.
Green highlight: The maximum value of each optimizer for five base networks.
Cyan highlight: The best result for each base network (the highest result of mean accuracy or weighted IoU together with the highest result of crack accuracy or background accuracy).

Table 2. Training parameters of each model in Table 1

No.	Base network, optimizer	Mini-batch size	Learning rate	Squared gradient decay factor
1	ResNet-18, SGDM	32	0.01	N/A
2	ResNet-18, RMSProp	28	0.00001	0.99
3	ResNet-18, Adam	32	0.00001	0.999
4	ResNet-50, SGDM	26	0.001	N/A
5	ResNet-50, RMSProp	24	0.00001	0.9
6	ResNet-50, Adam	26	0.00001	0.999
7	MobileNet-v2, SGDM	30	0.0003	N/A
8	MobileNet-v2, RMSProp	28	0.000003	0.9
9	MobileNet-v2, Adam	30	0.000003	0.999
10	Xception, SGDM	24	0.0003	N/A
11	Xception, RMSProp	28	0.000003	0.999
12	Xception, Adam	26	0.000003	0.9
13	Inception-ResNet-v2, SGDM	6	0.001	N/A
14	Inception-ResNet-v2, RMSProp	6	0.00001	0.9
15	Inception-ResNet-v2, Adam	4	0.00001	0.9

Table 3. The rank of recommended models

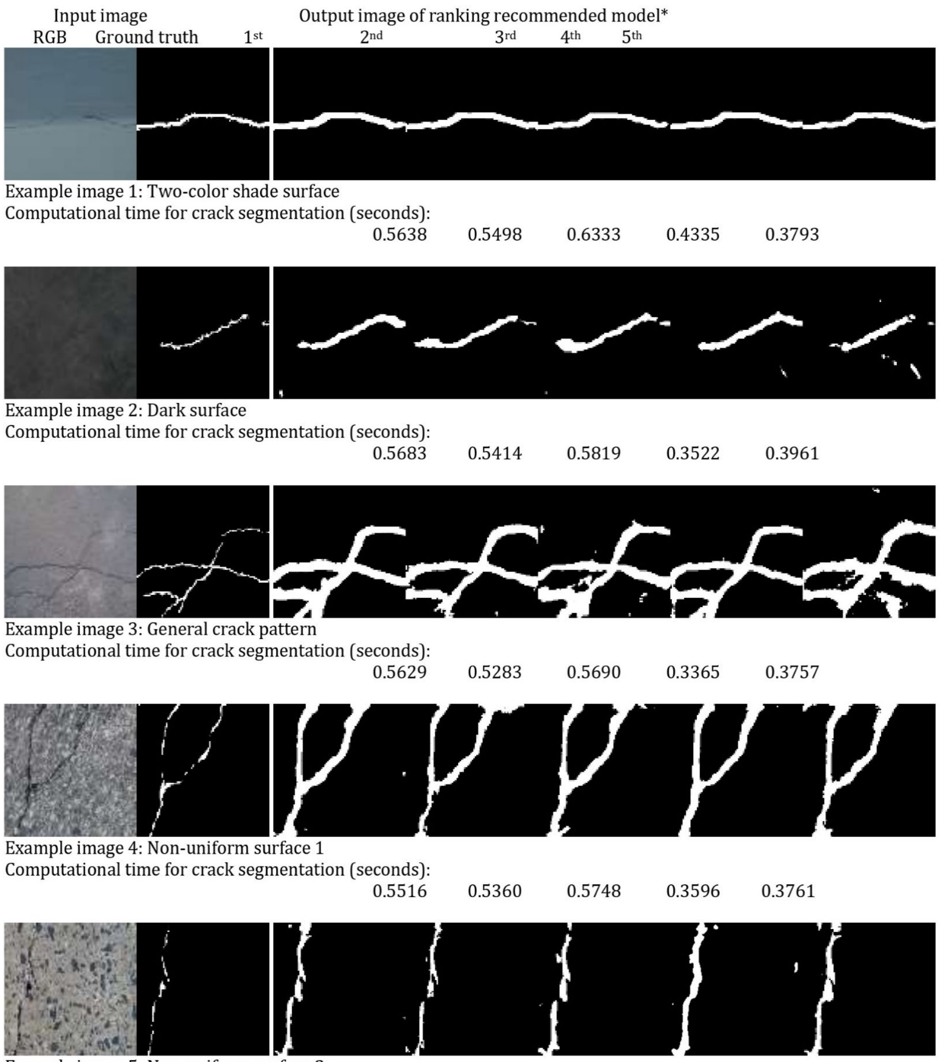
Rank	Base network, optimizer	Model size (MB)	Description
1	ResNet-50, Adam	142	Highest crack accuracy and mean accuracy
2	Xception, RMSProp	84.2	Highest background accuracy and weighted IoU
3	ResNet-50, SGDM	141	High values of crack accuracy, background accuracy, mean accuracy, and weighted IoU
4	ResNet-18, SGDM	58.4	High all measured values but less than the results of ResNet-50 base network with SGDM optimizer
5	MobileNet-v2, RMSProp	9.50	High all measured values but less than the results of ResNet-18 base network with SGDM optimizer

Table 4. Performance summarization of the best five models of DeepLab v3+ in crack segmentation

Rank	Base network, optimizer	Mean accuracy	Crack accuracy	Background accuracy	Weighted IoU
1	ResNet-50, Adam	<u>**0.95048**</u>	<u>**0.96944**</u>	0.93151	0.90977
2	Xception, RMSProp	**0.94716**	0.94789	<u>**0.94642**</u>	<u>**0.92507**</u>
3	ResNet-50, SGDM	0.94960	0.95502	**0.94417**	**0.92291**
4	ResNet-18, SGDM	**0.94616**	0.95001	**0.94230**	**0.92063**
5	MobileNet-v2, RMSProp	0.94192	0.94354	**0.94031**	**0.91814**

Note:
Bold: The highest values in each base network.
<u>Underline</u>: The highest values of all base networks.
Green highlight: The maximum value of each optimizer for five base networks.

Figure 5 shows the example image results of the five recommended models. The first and second images are examples of two-color shade and dark surfaces, respectively. The third and fourth images describe some examples of crack patterns, while the fourth and fifth images show examples of non-uniform surfaces. The computational time of crack segmentation model in each example is shown in seconds below the image in Figure 5. The visualized image results reveal good identification of the crack pattern under various surface conditions. However, the prediction results usually segment a larger area of crack, which could be investigated further for future improvement.



Example image 1: Two-color shade surface
Computational time for crack segmentation (seconds):
0.5638 0.5498 0.6333 0.4335 0.3793

Example image 2: Dark surface
Computational time for crack segmentation (seconds):
0.5683 0.5414 0.5819 0.3522 0.3961

Example image 3: General crack pattern
Computational time for crack segmentation (seconds):
0.5629 0.5283 0.5690 0.3365 0.3757

Example image 4: Non-uniform surface 1
Computational time for crack segmentation (seconds):
0.5516 0.5360 0.5748 0.3596 0.3761

Example image 5: Non-uniform surface 2
Computational time for crack segmentation (seconds):
0.5880 0.5345 0.5636 0.3571 0.3788

Figure 5. Example image results of the recommended models
Note: * 1st rank: ResNet-50, Adam; 2nd rank: Xception, RMSProp; 3rd rank: ResNet-50, SGDM; 4th rank: ResNet-18, SGDM; 5th rank: MobileNet-v2, RMSProp

4. DISCUSSION

This research prioritizes the consideration of hardware resource constraints to enable individuals with limited budgets to participate in this field and pursue future advancements. Therefore, the conditions of network architectures that can be trained using limited GPU hardware resources, i.e., a single GPU; NVIDIA® GeForce RTX™ 3070 with 8GB of GPU memory, are also related to the model size and mini-batch size. Large network architecture like DeepLab v3+ with Inception-ResNet-v2 base network, i.e., with a model size larger than 200 MB and total number of layers more than 800, practically require a high memory GPU for training and cannot let the mini-batch size go so high due to the memory constraints (a larger mini-batch size usually requires more memory). Therefore, even though the DeepLab v3+ with Inception-ResNet-v2 base network has the highest number of layers and learnable parameters, as shown in Table 1, it could not provide the highest accuracy because of the limitation in mini-batch size during the model training. As the mini-batch size of Inception-ResNet-v2 base network that can be run without the "out of memory error" is smaller than the others, as shown in Table 2 (not much variation in mini-batch size can be experimented with). This might have led the overall result to be lower than the other models.

For the five recommended models, based on the summarized information related to the models, shown in Table 5, the first three ranks had a similar number of layers, and have a number of layers and a number of learnable parameters higher than the fourth and the fifth ranks without much difference in mini-batch size. The higher complexity of the model may enable the first three ranks to achieve superior performance compared to the remaining two ranks. The first and the third ranked models had the highest number of layers and number of learnable parameters from the five recommended models. These factors could contribute to the models achieving the highest and second-highest crack accuracy and mean accuracy, as demonstrated in Table 4. However, the increase in mini-batch size of the second ranked model compared to the first and the third ranks might assist in the better performance, even with a smaller number of learnable parameters. Therefore, the overall performance of the second ranked model is high particularly in obtaining the highest background accuracy and weighted IoU. However, the top three ranked models could be flexibly used depending on the desired model size, with the second ranked model offering a smaller model size. When examining the fourth and fifth ranked models, in Table 5, while the fourth ranked model had fewer layers, it exhibited greater complexity due to its larger number of learnable parameters and model size. This could lead to overall better performance, as observed in Table 4.

For the computational time, the results of each image example shown in Figure 5 were summarized and compared in Figure 6. Considering the model architecture, specifically the number of layers and learnable parameters and the model size, the results of computational time for crack segmentation could be grouped into the first three ranked models with a computational time exceeding 0.5 seconds and the last two ranked models with computational times less than 0.5 seconds, as illustrated in Figure 6. However, the computational time for crack segmentation can also be influenced by other factors, resulting in some variations within each group.

Based on the model sizes listed in Table 3, all five recommended models can be executed on standard desktop or laptop computers. However, for IoT applications, the performance of the Raspberry Pi series is generally recommended. A smaller model, such as the fifth ranked model, might be suitable for practical implementation. Many Arduino and ESP32 series devices may struggle to run these models due to insufficient memory.

In general, if the model size is not a condition, the first ranked model (ResNet-50 based network with Adam optimizer) would be the best choice for this application and training for the new images of specific structures under constrained resources. However, if the model size needs to be considered, the second and fourth ranked models might be the solution. For mobile and embedded applications, the fifth ranked model of MobileNet-v2 base network with RMSProp optimizer is recommended.

The research results will enhance and provide guidance for the utilization of DeepLab v3+ on crack segmentation and new image training on diverse concrete structures using limited hardware resources.

Table 5. The summarized model information of the five recommended models

Rank	Base network, optimizer	Mini-batch size	Learning rate	Number of learnable parameters	Number of layers	Model size (MB)
1	ResNet-50, Adam	26	0.00001	43,980,180	206	142
2	Xception, RMSProp	28	0.000003	27,638,052	205	84.2
3	ResNet-50, SGDM	26	0.001	43,980,180	206	141
4	ResNet-18, SGDM	32	0.01	20,607,636	100	58.4
5	MobileNet-v2, RMSProp	28	0.000003	6,784,276	186	9.50

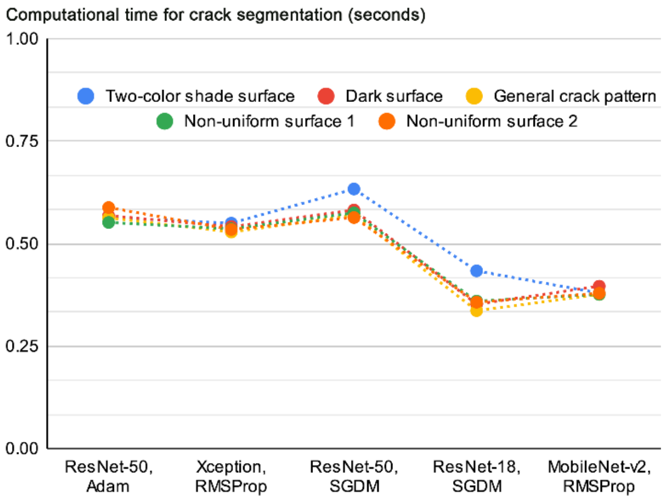


Figure 6. The Comparison of the computational time for crack segmentation of the examples in Figure 5

5. CONCLUSION

In this paper, the performance of DeepLab v3+ was compared with alternative base networks, i.e., Inception-ResNet-v2, Xception, ResNet-50, MobileNet-v2, and Res-Net-18. This experiment was conducted to find the best parameters of mini-batch size, learning rate, and squared gradient decay factor, for training the networks employing limited resource with three optimization algorithms, i.e., SGDM, RMSProp and Adam. The evaluation was conducted focusing on the accuracy and the weighted IoU. The accuracy of crack class and background class, the mean accuracy, and the weighted IoU were investigated to determine the best result for each base network of DeepLab v3+. Then the best results were analyzed in terms of mean accuracy, class accuracy, weighted IoU, and model size to recommend the suitable model and training parameters for resource-constrained applications and training environment for a broader and more diverse range of infrastructure applications. The first rank of recommended model was DeepLab v3+ network based on ResNet-50 with Adam optimizer because of the highest crack accuracy and mean accuracy. If the model size is an issue to consider, the Xception base network with RMSProp optimizer and the ResNet-18 base network with SGDM optimizer are recommended. In addition, the results clearly show that the MobileNet-v2 base network with RMSProp optimizer is the smallest model and most suitable for mobile and embedded applications. It is hoped to conduct further study on the DeepLab v3+ network in various aspects in the future to enhance the maximum capability of this powerful and efficient network using limited processing resources.

REFERENCES

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer vision – ECCV 2018. Lecture note in computer science, volume 11211* (pp. 833–851). Springer. https://doi.org/10.1007/978-3-030-01234-2_49

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1800–1807). IEEE. https://doi.org/10.1109/CVPR.2017.195

Cui, X., Wang, Q., Dai, J., Li, S., Xie, C., & Wang, J. (2022). Pixel-level intelligent recognition of concrete cracks based on DRACNN. *Materials Letters*, *306*, Article 130867. https://doi.org/10.1016/j.matlet.2021.130867

Du, J. (2019). The frontier of SGD and its variants in machine learning. *Journal of Physics: Conference Series*, *1229*, Article 012046. https://doi.org/10.1088/1742-6596/1229/1/012046

Fu, H., Meng, D., Li, W., & Wang, Y. (2021). Bridge crack semantic segmentation based on improved Deeplabv3+. *Journal of Marine Science and Engineering*, *9*(6), Article 671. https://doi.org/10.3390/jmse9060671

Han, C., Ma, T., Huyan, J., Huang, X., & Zhang, Y. (2022). CrackW-Net: A novel pavement crack image segmentation convolutional neural network. *IEEE Transactions on Intelligent Transportation Systems*, *23*(11), 22135–22144. https://doi.org/10.1109/TITS.2021.3095507

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778). IEEE. https://doi.org/10.1109/CVPR.2016.90

Kheradmandi, N., & Mehranfar, V. (2022). A critical review and comparative study on image segmentation-based techniques for pavement crack detection. *Construction and Building Materials*, *321*, Article 126162. https://doi.org/10.1016/j.conbuildmat.2021.126162

Kim, J., Shim, S., Cha, Y., & Cho, G.-C. (2021). Lightweight pixel-wise segmentation for efficient concrete crack detection using hierarchical convolutional neural network. *Smart Materials and Structures*, *30*(4), Article 045023. https://doi.org/10.1088/1361-665X/abea1e

Lau, S. L. H., Chong, E. K. P., Yang, X., & Wang, X. (2020). Automated pavement crack segmentation using U-Net-based convolutional neural network. *IEEE Access*, *8*, 114892–114899. https://doi.org/10.1109/ACCESS.2020.3003638

Li, H., Wang, W., Wang, M., Li, L., & Vimlund, V. (2022). A review of deep learning methods for pixel-level crack detection. *Journal of Traffic and Transportation Engineering (English Edition)*, *9*(6), 945–968. https://doi.org/10.1016/j.jtte.2022.11.003

Liu, J., Yang, X., Lau, S., Wang, X., Luo, S., Lee, V. C.-S., & Ding, L. (2020). Automated pavement crack detection and segmentation based on two-step convolutional neural network. *Computer-Aided Civil and Infrastructure Engineering*, *35*(11), 1291–1305. https://doi.org/10.1111/mice.12622

Liu, Y., Yao, J., Lu, X., Xie, R., & Li, L. (2019). DeepCrack: A deep hierarchical feature learning architecture for crack segmentation. *Neurocomputing*, *338*, 139–153. https://doi.org/10.1016/j.neucom.2019.01.036

Mei, Q., & Gül, M. (2020). A cost effective solution for pavement crack inspection using cameras and deep neural networks. *Construction and Building Materials*, *256*, Article 119397. https://doi.org/10.1016/j.conbuildmat.2020.119397

Mohan, A., & Poobal, S. (2018). Crack detection using image processing: A critical review and analysis. *Alexandria Engineering Journal*, *57*(2), 787–798. https://doi.org/10.1016/j.aej.2017.01.020

Nguyen, N. H. T., Perry, S., Bone, D., Le, H. T., & Nguyen, T. T. (2021). Two-stage convolutional neural network for road crack detection and segmentation. *Expert Systems with Applications*, *186*, Article 115718. https://doi.org/10.1016/j.eswa.2021.115718

Nguyen, T.-G., Do, T.-L., Nguyen, T.-N., & Nguyen, N.-N. (2024). Semantic segmentation of cracks using DeepLabv3+. In J. N. Reddy, C. M. Wang, V. H. Luong, & A. T. Le (Eds.), *Proceedings of the Third International Conference on Sustainable Civil Engineering and Architecture (ICSCEA 2023)* (pp. 1539–1546). Springer. https://doi.org/10.1007/978-981-99-7434-4_165

Pu, R., Ren, G., Li, H., Jiang, W., Zhang, J., & Qin, H. (2022). Autonomous concrete crack semantic segmentation using deep fully convolutional encoder–decoder network in concrete structures inspection. *Buildings*, *12*(11), Article 2019. https://doi.org/10.3390/Buildings12112019

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4510–4520). IEEE. https://doi.org/10.1109/CVPR.2018.00474

Su, C., & Wang, W. (2020). Concrete cracks detection using convolutional neural network based on transfer learning. *Mathematical Problems in Engineering*, *2020*(1), Article 7240129. https://doi.org/10.1155/2020/7240129

Su, H., Wang, X., Han, T., Wang, Z., Zhao, Z., & Zhang, P. (2022). Research on a U-Net bridge crack identification and feature-calculation methods based on a CBAM attention mechanism. *Buildings*, *12*(10), Article 1561. https://doi.org/10.3390/buildings12101561

Sun, X., Xie, Y., Jiang, L., Cao, Y., & Liu, B. (2022). DMA-Net: DeepLab with multi-scale attention for pavement crack segmentation. *IEEE Transactions on Intelligent Transportation Systems*, *23*(10), 18392–18403. https://doi.org/10.1109/TITS.2022.3158670

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017). Inception-v4, Inception-ResNet and the impact of residual connections on learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1), 4278–4284. https://doi.org/10.1609/aaai.v31i1.11231

Talab, A. M. A., Huang, Z., Xi, F., & HaiMing, L. (2016). Detection crack in image using Otsu method and multiple filtering in image processing techniques. *Optik*, *127*(3), 1030–1033. https://doi.org/10.1016/j.ijleo.2015.09.147

Xie, Z., Lu, Q., Guo, J., Lin, W., Ge, G., Tang, Y., Pasini, D., & Wang, W. (2024). Semantic segmentation for tooth cracks using improved DeepLabv3+ model. *Heliyon*, *10*(4), Article e25892. https://doi.org/10.1016/j.heliyon.2024.e25892

Xu, G., Yue, Q., & Liu, X. (2023). Deep learning algorithm for real-time automatic crack detection, segmentation, qualification. *Engineering Applications of Artificial Intelligence*, *126*(Part C), Article 107085. https://doi.org/10.1016/j.engappai.2023.107085

Yang, L., Huang, H., Kong, S., & Liu, Y. (2023). A deep segmentation network for crack detection with progressive and hierarchical context fusion. *Journal of Building Engineering*, *75*, Article 106886. https://doi.org/10.1016/j.jobe.2023.106886

Zhou, Z., Zheng, Y., Zhang, J., & Yang, H. (2023). Fast detection algorithm for cracks on tunnel linings based on deep semantic segmentation. *Frontiers of Structural and Civil Engineering*, *17*(5), 732–744. https://doi.org/10.1007/s11709-023-0965-y