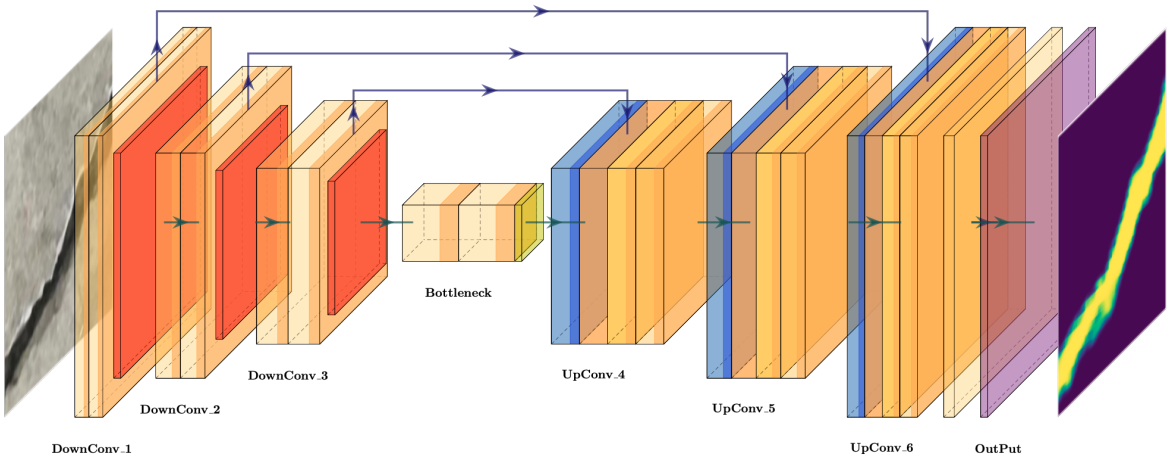
CHALMERS
UNIVERSITY OF TECHNOLOGY



AI-enhanced Algorithm for Structural Health Monitoring

An Image-based Concrete Crack Detection Method Using Convolutional Neural Networks

Master's thesis in Structural Engineering and Building Technology

XI LUO
JIA GUO

DEPARTMENT OF ARCHITECTURE AND CIVIL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

AI-enhanced Algorithm for Structural Health Monitoring

An Image-based Concrete Crack Detection Method Using Convolutional Neural Networks

XI LUO
JIA GUO

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Architecture and Civil Engineering
Division of Structural Engineering
Concrete Structures Research Group Chalmers University of Technology
Gothenburg, Sweden 2021

AI-enhanced Algorithm for Structural Health Monitoring
An Image-based Concrete Crack Detection Method Using Convolutional Neural Networks
XI LUO
JIA GUO

Supervisor: Kamyab Zandi, Department of Architecture and Civil Engineering
Examiner: Kamyab Zandi, Department of Architecture and Civil Engineering

Master's Thesis 2021
Department of Architecture and Civil Engineering
Division of Structural Engineering
Concrete Structures Research Group Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A convolutional neural network architecture for image segmentation.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

AI-enhanced Algorithm for Structural Health Monitoring
An Image-based Concrete Crack Detection Method Using Convolutional Neural Networks
XI LUO
JIA GUO
Department of Architecture and Civil Engineering
Chalmers University of Technology

Abstract

The thriving of image-based damage detection methods for structure health monitoring especially the application of UAVs for structure inspection has been a trend in the recent years. The concept of Digital Twin is to build a living digital representation for structures which requires a very fast data processing procedure. This paper proposes a CNN-based crack detection method that can recognize and extract cracks from photos of concrete structures, which can enhance the data processing for Digital Twin. The algorithm consists of two subsequent procedures, classification and segmentation, achieved by two convolutional neural networks respectively. First, full images are divided into patches and classified as positive and negative. Then, those sub-images classified as positive where cracks are visible are further processed by the image segmentation procedure to obtain the pixel level shapes of the cracks. For the classification part, the performance of transfer learning models based on pre-trained VGG16, Inception V3, MobileNet and DenseNet169 is compared with different classifier. Finally, the CNN based on MobileNet was trained with 30,000 training images and can reach 97% testing accuracy and 0.96 F1 score on testing image. For the segmentation part, different neural networks based on the elegant U-net architecture are built and tested. The models are trained with 3840 crack images and annotated ground truth and compared quantitatively and qualitatively. The model with the best performance can reach 88% sensitivity on test data set. The combination of the classification and segmentation neural networks can achieve an image-based crack detection method with high efficiency and accuracy. The algorithm can process any full image size as input. Compared with most machine learning based crack detection algorithms using sub-image classification, a relatively larger patch size is used in this paper and in this way the classification is more robust and accurate. On the other hand, the negative area in the full image will not be concerned in the segmentation procedure and this fact not only saves a lot of computational power but also significantly increases the accuracy compared to the segmentation performed on full images.

Keywords: Digital Twin, Crack Detection, Convolutional Neural Network, Computer Vision, Image Classification, Semantic Segmentation.

Acknowledgements

It is a challenge for both of us to conduct a master thesis related to the topic of computer vision and machine learning but we are so grateful that we got this chance to learn and practice the most advanced field of structural health monitoring. We want to express our most sincere appreciation to Kamyab Zandi, who has been supervising our project for the the past half year and providing us advises to help us carry out this study in a right way. We are grateful to Brosamverkan for the financial support so that we can have a chance to have a drone and test our method. Our colleague Henrik Waldäng collected photos of damaged concrete structures and these are important source data for our neural networks. We also appreciate his contribution.

Chalmers is an amazing place and both of us enjoyed the past two years of our master program. We have met interesting colleagues and teachers and made new friends. It's a totally different life in Nordic countries and the pandemic of Covid-19 has made life more difficult. Even so this experience will still be one of the best memories in our life. Thank you, Chalmers.

Xi Luo and Jia Guo, Gothenburg, June 2021

Contents

List of Figures

List of Tables

1
Introduction

1.1 Background and Problem Description

Structure Health Monitoring (SHM) refers to the discipline of regular inspection or real-time monitoring of civil structures[1]. The information of the structure is collected by human inspection or multiple kinds of sensors, based on which the safety state of the structure will be assessed and potential risks can be recognized at a early stage. In this way, a large amount of costs can be saved and more importantly, public safety can be ensured.

The concept of Digital Twin (DT), which is first proposed by NASA in 2010, serves as a digital representative of a physical object. It is a multi-physics and multi-scale simulation that reflects the real-time state of the corresponding object, based on both historical data and real-time sensor data.[2]. In the context of Structure Health Monitoring, Digital Twin serves as a living digital model of the real structural, providing structure performance predictions by processing varieties of data collected by sensors[3].

Figure 1.1 illustrates the main procedures of Digital Twin for SHM. First, information of the structure is collected by devises like a camera or embedded sensors. Nowadays the thriving of Unmanned Aerial Vehicles (UAV) has provided a more flexible method to inspect civil structures. These data, in the form of digital images or signals form example , will be process and structural damages will be recognized and located automatically. Finally the digital representative of the structural will be updated with the detected damages and structural behaviour will be analyzed.

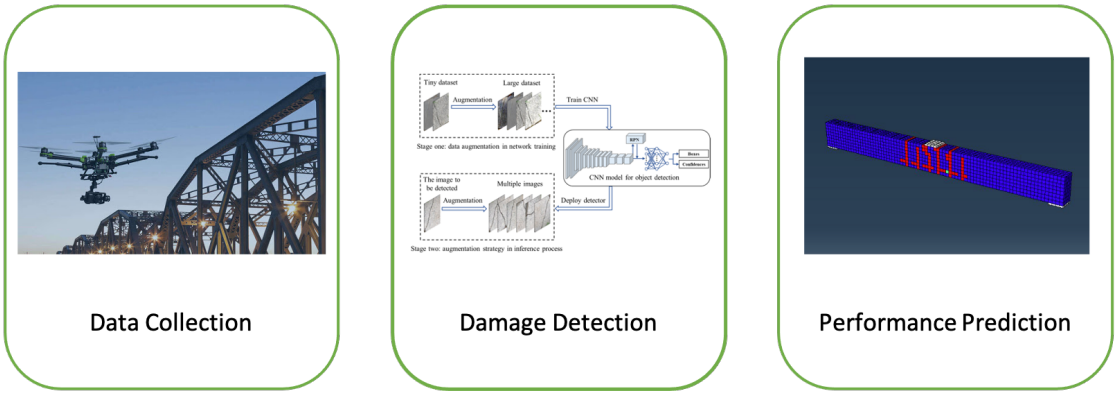


Figure 1.1: *Schematic diagram of Digital Twin for SHM*

The application of DT in the SHM field provides a efficient and low-cost strategy. However, integrating both time and space information, DT demands high level information analytic and computational power to support its up-to-date simulation. Nowadays, with the fast development of information transmission technology like 5G network and cloud computing, the concept of DT provides a grand new possibility for SHM and arouse attention in the academia[4].

For concrete structures, cracks are a kind of typical signs of performance deterioration. Detection of cracks is a important part of the SHM for concrete structures. Traditional methods of crack detection are usually done manually. Nowadays, non-contact sensors like high-speed cameras and Unmanned Aerial Vehicles (UAV) are widely used in the SHM field. Crack detection based on image data has been proved to be a more efficient option [5]. Strategies like image processing method and machine learning algorithms have been developed for this purpose, among which Convolutional Neural Network (CNN) has recently become a popular class of Deep Learning algorithms for image recognition problems[6].

1.2 Goals and Objectives

This thesis investigates the second procedure of the Digital Twin work flow shown in Figure 1.1 and the topic is the application of Artificial Intelligence algorithm in the SHM field. Specifically, the aim is to establish a well-functioning image-based crack detection method using CNN models, which is expected to recognize cracks in an image and locate damage on the structure surface. Moreover, the algorithm should also be robust to noise data collected from rough concrete surface to reduce the false identification rate.

The outcome of this algorithm will be a part of the creation of a Digital Twin of a concrete structure, which will provide practical information that can be used in the simulation of structure performance. Many further applications can also be built on this highly autonomous crack detection procedure. In the long term, this project can contribute to the promotion of the non-contact sensors and Digital Twin in the SHM field. Regarding the aspect of social effect, these advanced technologies are expected to provide a more cost-efficient and sustainable way to perform structure maintains, which eventually benefits both the industry and ecology.

1.3 Scope and Limitation

This thesis focuses on the topic of concrete structure health monitoring and the proposed algorithm is designed for the specific task of cracks detection for concrete structure surfaces. Raw image data collected from both concrete structure specimens and real concrete structural elements will be used for training and testing of the proposed networks. For data-driven algorithms like CNN, the database plays an important role in the network training and testing procedure. The raw data collection is beyond the scope of this thesis and existing databases established previously in other projects will be used.

The proposed image-based crack detection algorithm is supposed to serve as a part of the Digital Twin program and the output of the algorithm will be reflected in the digital twin simulation of the physical concrete structural. The function of this algorithm is limited to identify and locate the crack in a image of a concrete surface. Absolute accuracy is hard to achieve and the applications of CNN in SHM community have never reached 100% of accuracy either [6]. The proposed models in this thesis aim at a faster and fully autonomous algorithm with a acceptable robustness and a better applicability in the field inspection setting.

1.4 Literature Review

In the SHM field, image based crack detection using CNN algorithm has been proved to be a feasible method[6]. A literature review is conducted first to study the advanced techniques for image-based crack detection using CNN.

Many researches has studied the topic of image-based crack detection as a sub-image classification problem. Zhang et al. [7] proposed a ConvNet model trained with more than 500 pavement images, which can recognize the presence of road cracks in a square image patch sized 99×99 pixels. Using a similar strategy, Kim et al. [8] proposed an transfer learning network based on pre-trained R-CNN to detect cracks on concrete bridge. Sobel edge detection algorithm was also used in that study to quantitatively measure cracks. Inspired by the famous CNN model AlexNet for image classification, Kim and Cho establish a algorithm that can classify sub-images sized 227×227 into 4 classes (cracks, structural joints, plants and intact surface). The multi-class model and the large window size significantly increased the identification accuracy and by using a overlap window the region of cracks is narrowed. All the researches mentioned above used the sliding window method which first divide a full image of concrete surface into sub-images and perform image classification on the sub-images. Thus the sub-images with cracks will be selected and the area of damage is obtained. But this method has a resolution of sub-image level and provides no quantitative information about the cracks like the width and the length. Further, when small sub-image size is used, meaning the amount of the input data is smaller, the accuracy and robustness of CNN classifiers also decrease.

Another direction of image-based crack detection is image segmentation, which can reach a pixel-level resolution. Lin et al. [9] proposed a CNN model based on the famous U-net architecture to perform full-image segmentation. Attention gate technique is employed in the model to improve the performance on small object detection task. Zhang et al. [10] investigated different architectures based on U-net and compare their performance on 256×256 sub-images, and generalized loss functions are used in the training of the models. The image segmentation strategy can reach better resolution of the target cracks then the sub-image classification, but it also costs considerably more computational power to finish the task, especially for full image processing. Another problem is that cracks are usually extremely small object in a full image of a concrete surface. For this type of small target segmentation, the training of CNN models becomes unstable.

1.5 Approach and Outline

This thesis proposes a combination of image classification and segmentation CNN models to reach a algorithm with high accuracy and efficiency. First, full images are divided into patches and classified as positive and negative. Then, those sub-images classified as positive where cracks are visible are further processed by the image segmentation procedure to obtain the pixel level shapes of the cracks. The combined algorithm has advantages in many aspect. Firstly, it can process any full image size as input, which means there is no limitation for data collection devices. Secondly, compared with most machine learning based crack detection algorithms using sub-image classification, a relatively larger patch size can be used and the classification is more robust and accurate. Lastly, the negative area in the full image will not be concerned in the segmentation procedure and this fact not only saves a lot of computational power but also fix the small target segmentation issue and significantly increases the accuracy compared to the segmentation performed on full images. This thesis consists of six chapters

1. **Introduction**: A brief overview of this thesis, including the background of the SHM and Digital Twin concepts. Introduce the problem needs to be solved, the objectives, scopes, relevant researches and approaches of this master thesis project.

2. **Convolutional Neural Network for Structure Health Monitoring**: Present the basic theory of CNN algorithm and it's applications in the SHM field. Several advanced techniques that has been developed to optimize CNN algorithms for computer vision problems are also introduced in this chapter.

3. **Image Classification**: This chapter records all the hyper-parameter adjusting process and the techniques has been used for getting a better accuracy. Based on the final prediction results, one best model will be chosen and used in the algorithm.

4. **Image Segmentation**: CNN models based on the elegant U-net architecture for image segmentation tasks are built and trained with pixel-level annotated crack sub-images. The model are evaluated quantitatively and qualitatively.

5. **Experiment**: Photos of cracks are taken by a drone in Chalmers campus, from a damage wall in the structure lab and a concrete specimen. The photo are used to test the proposed algorithm and the crack detection result is illustrated in this chapter.
6. **Conclusion**: Presents a discussion on the performance of proposed model and a summary of this thesis, including the conclusion of this study and possible improvements. The possible application of the outcome and the prospect of the future study is also discussed.

2

Convolutional Neural Network for Structure Health Monitoring

2.1 Application of CNN in SHM field

Cracks on the concrete surface are the early sign of concrete degradation, as they can accelerate the corrosion process and propagate to severer concrete defects in the later stage. Therefore, early crack detection has always been an essential task in structural health monitoring and ritual observation object on an inspection basis. Nowadays, robots or unmanned-aerial vehicles can partial the collecting process, leaves only the image-based crack detection work for the inspector. However, manually selecting the crack in the concrete image is very time-consuming and labor-intensive, which will exaggerate when the image database becomes large. Image enhancement technique can highlight the cracked parts to ease the crack selecting process difficulty, thus always show up as data post-processing step in image processing steps. The traditional image processing technique, known as the automated crack detection process based on various edge detecting algorithms, can identify the cracked parts within the concrete images without anthropogenic intervention. However, it still needs a final check about the processed results. Unlike the traditional image processing technique, the deep learning method can predict the results without review, just with some labeled data for full training, which has become the hot trend of this time.

The image processing technique (IPT) in the crack detection field generally indicates the automated crack prediction process merely done by a mathematical algorithm. The flowchart 2.1 below shows the work mechanism pipeline. Various edge detector works exceptionally when the concrete image contains not too much noise at edge detection step. The next step is to apply some image enhancement techniques to increase the clarity of the edge feature. The final step is to extract the cracked features by image segmentation techniques such as binarization or thresholding. Based on a study done by Abdel-Qader et al.[11] who systemically studied the Fast Haar transform, Fourier transforms, Sobel filter, and Canny filter, concluded that conclusion that the fast Haar transform has the top performance with 86% accuracy, followed with Canny filter. The following study involved different religion-based crack detection techniques like support vector machine and the penetration model developed in 2008 by Yamaguchi et al.[12] Nashikawa et al.[13] fought a way to quantify the crack width by the illuminance distribution in one image in 2012. However, the traditional IPTs have the mutual drawback of lacking the resilience

for image noise like the clarity and lighting condition. Lecun et al.[14] who applying the convolutional neuronal network to classify crack image brought an innovative way of crossing the hurdle traditional IPT remains. Sattar Dorafshan, Robert J et al. conducted a study in 2018 about comparing four edge detection methods in the frequency domain (Roberts, Prewitt, Sobel, and Laplacian of Gaussian) and two in the frequency domain (Butterworth and Gaussian), along with the AlexNet-based DCNN in different training condition.[15] The author claim that DCNN in the transfer learning model performs better than the IPT with higher sensitivity for the thinner crack from 0.1mm down to 0.04mm and the accuracy from 53-79% to 86% in a shorter computational time for the prediction process itself. By proposing a hybrid detection model at the end of this paper, this study revealed the promising future application of CNN in the crack detection field.

Figure 2.1: *The pipeline of IPT*

2.2 The development of CNN

The history of the convolutional neural network (CNN) development has three different stages. The first stage is the invention of CNN. After the inspiration from a biology experiment based on the visual cortex cell of cats, Hubel and Wiese developed the concept receptive field to describe the hierarchy transmission of stimuli in the Visual system in the 1960s [16]. This biology observation begot the incipient CNN called Neocognitron and coined by Fukushima in the 1980s [17]. As a hierarchical network like the biology cortex system, the Neocognitron has the properties that once done the training, the recognition process will not be affected by a slight deformation of input data or spatial change (Figure 2.2). The novelty of this research is the innovative imitation of how the information passing between cells or neurons. Back at that time, the unsupervised learning method was the choice. Although the Neocognitron can manage to train the higher layer forward, this learning process will strongly depend on the training pattern input, which will become the main reason for upgrading difficulty when the recognition task becomes complex.

LeCun, et el.[18] who applied a backpropagation algorithm improve the deficiency of the previous work. They trained the network known as LeNet-5 by using the convolutional filter to extract the features in the original image, then proceeding it forward, which has outstanding performance. Success in recognizing the handwritten digit by LeNet-5 aroused the intention of academia. Despite the improvement on training algorithm, the problems of deep connection such as vanishing gradient, overfitting still are challenges to be resolved.

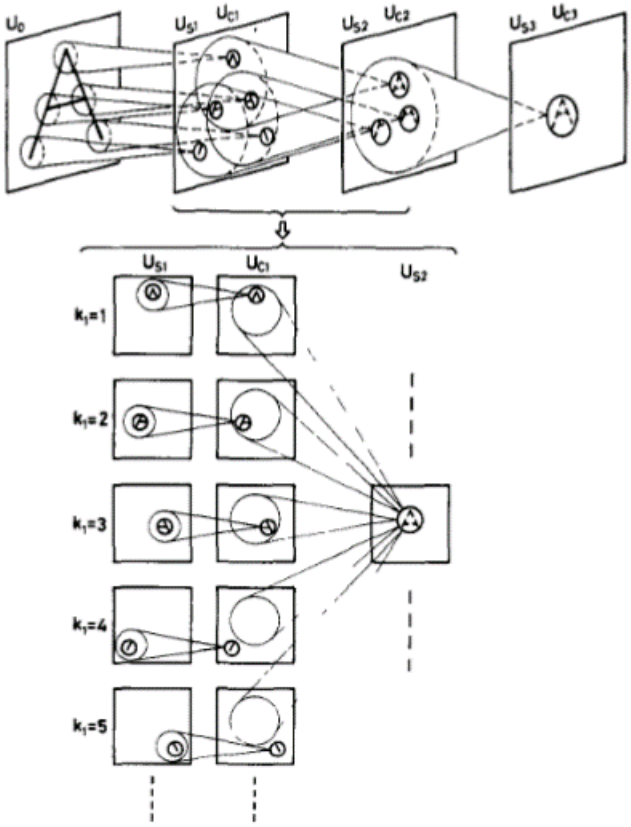


Figure 2.2: *The cell synaptic connections between cells, and the response of the cells after completion of the self-organization*

The fast growth of artificial neural network techniques over the past decades facilitates CNN development. For instance, Xavier, Yoshua, et al. brought up the normalized initialization scheme to counter the vanishing gradient issue.[19] With the image classification contest held in 2012, the champion Krizhevsky proposed the well-known AlexNet[20], achieved a test error rate of 15.3 %, which is half less than the runner-up, brought the field further heat. After AlexNet, the CNN research field was blooming. For instance, the development of VGG-Net proposed by Visual Geometry Group, GoogLeNet from Google, drove CNN to become more and more sophisticated for industry applications.

2.2.1 The architecture of CNN

2.2.1.1 The input layer

The input layer can represent the pixel value for one RGB image, often shows as a multidimensional tensor. If the network processes the RGB images have three channels, the input layer is a tensor with four dimensions. Generally, the first dimension indicates the image number, then the rest of the dimensions are the image width, image height, and image depth. After the input layer, the image will be pass to the overall convolutional neural network.

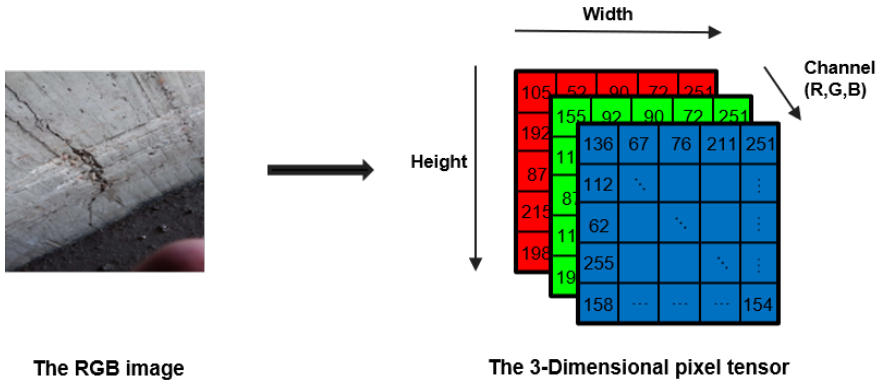


Figure 2.3: *The 3-Dimensional tensor for a RGB image*

2.2.1.2 The convolutional layer

The convolutional layer plays the most pivot part in the overall network by convolving the input layer then devolving it to the next layer. The convolutional filters, which have the learnable weights for detecting the features to represent the original image, did the actual convolution operation. The hyperparameter will also include stride and padding size. Stride is the number of pixels that each time the filter leaps, padding size is the complementary pixel added in a specified way for purposes like confining output range.

In most situations, the non-linear activate function will follow the convolutional layer to empower the network to learn non-linear characteristics.

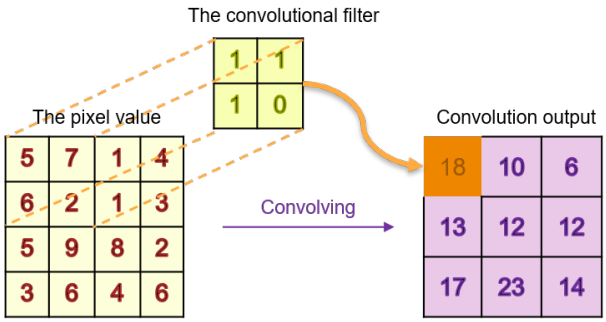


Figure 2.4: *Visualization of convolutional process with 2*2 fliter*

2.2.1.3 The pooling layer

The pooling layer performs the sampling to reduce the output size. The pooling layer only keeps the maximum value among pixels is called the max-pooling layer, whereas it will keep the average pixel value in the average-pooling layer. In most cases, the pooling layer can help to ease the cramming data flow and prevent overfitting.

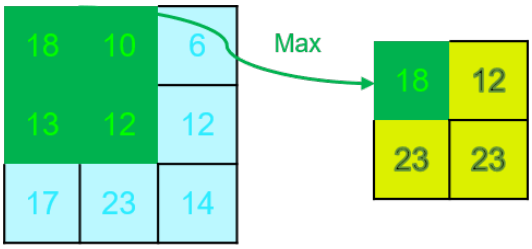


Figure 2.5: *Visualization of max-pooling process with 2*2 fliter and stride 1*

2.2.1.4 The fully connection layer

The fully connected layer is a typical multi-layer perceptron neural network that will connect the flattened convolutional layer to the final output class. Generally, the last fully connected layer is the output layer to make the final prediction: the probability of each target class or the actual prediction value.

2.2.2 The hyper-parameter

2.2.2.1 The loss function

The loss function, also known as the cost function, plays a vital role in the training process since the overall goal is to reduce the loss function. The loss will be the key value for updating the learnable parameters in networks like weights and bias. Generally, the customized loss function is more popular for the actual task. Based on three different task categories, this paper will list the most commonly used loss function below.

1. The Regression Loss
    - The Mean Squared Loss

      The mean squared loss is also known as L2 loss, calculated by the mean squared error between the truth and predicted value. This loss function will always give a positive value where zero means the perfect prediction.

      $$MSE = (y - f(x))^2 \tag{2.1}$$

    - The Mean Absolute Loss

      The mean absolute loss is the distance between the actual value and the predicted value. Be Known as L1 loss, which is more robust to resist the outliers compare with L2 loss.

      $$MAE = |y - f(x)| \tag{2.2}$$

    - The Huber Loss

      The Huber loss is a step function that synthesizes the benefits of both MSE and MAE. Thresholding the loss value by a factor δ, the loss smaller

than δ will be calculated as MSE otherwise MAE.

$$L_\delta = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{if } |y - f(x)| \leq \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{2.3}$$

2. The Binary Classification Loss Function

- The Binary Cross Entropy Loss
  The term entropy describes the uncertainty of one system or state where higher entropy usually indicates higher randomness.

$$S = \begin{cases} -\int p(x) \log p(x) dx & \text{if } x \text{ is continuous} \\ -\sum_x p(x) \log p(x) & \text{if } x \text{ is discrete} \end{cases} \tag{2.4}$$

  If the prediction is binomial, the entropy function will turn into:

$$L = -y * \log(p) - (1 - y) * \log(1 - p) \tag{2.5}$$

  If the binomial distribution is also mutex, means y can only be 0 or 1, will generate the definition of cross-entropy:

$$L = \begin{cases} -\log(1 - p) & \text{if } y = 0 \\ -\log(p) & \text{if } y = 1 \end{cases} \tag{2.6}$$

- The Hinge loss
  Hinge loss is suitable for the support vector machine, where the label y can only be -1 and 1 so that if the predicted value has low confidence to be the ground truth will also be penalized.

$$L = max(0, 1 - y * f(x)) \tag{2.7}$$

3. The Multi-class Classification Loss Function

- Multi-Class Cross Entropy Loss
  When the classification has multiple targets, the cross-entropy loss needs expansion. With one-hot labeled vector as target y_i, one can write the loss function as:

$$L(x_i, y_i) = -\sum_{j=1}^{c} y_{ij} * \log p_{ij} \tag{2.8}$$

  where y_i is a one-hot encoded vector, and only when i_{th} element in class j, $y_{ij} = 1$, otherwise is zero:

$$y_i = (y_{i2}, y_{i2}, ..., y_{ic}) \tag{2.9}$$

- The KL-Divergence
  The KL-Divergence is a mathematical expression for evaluating two different distribution groups. The KL-Divergence equal to zero when two distribution is identical.

$$D_{KL}(P\|Q) = \begin{cases} -\int p(x)\log\frac{Q(x)}{P(x)}dx = \int p(x)\log\frac{P(x)}{Q(x)}dx & \text{if the distribution is continuous} \\ -\sum_x p(x)\log\frac{Q(x)}{P(x)} = \sum_x p(x)\log\frac{p(x)}{q(x)} & \text{if the distribution is discrete} \end{cases} \tag{2.10}$$

2.2.2.2 The optimizing algorithm

The optimizing algorithm defines the network learning process. The development of the optimizing algorithm went through several stages, which can be represented by different optimizes.

1. Gradient Descent

   Gradient descent is the most fundamental updating algorithm across all types of machine learning models. By subtracting the previous weights with the derivative of itself in the loss function, the network updates after each training epoch toward the loss-dropping direction.

$$w'_{ij} = w_{ij} - \eta\nabla L(w)_{ij} \tag{2.11}$$

   Where η is the learning rate.

2. Stochastic gradient descent

   The stochastic gradient descent optimizing method is the gradient descent method with little reform. It updates the weights by using the loss over mini-batch input instead of the whole dataset.

$$w'_{ij} = w_{ij} - \eta\nabla L_{batch}(w)_{ij} \tag{2.12}$$

   In the later section, the notation g_t will be adapted to represent factor $\nabla L_{batch}(w)_t$.

3. Stochastic gradient descent with momentum

   Both previous methods only use the weight gradient to train the network and might lead the training process to impasse if the loss landscape becomes "flat" as there has no weight gradient for the deduction. This phenomenon is also known as the saddle point where some dimension in loss landscape has minimum but not the global minimum place. The stochastic gradient descent with momentum was ca-med up to resolve this insufficiency. The "momentum" is the gradient of the weights accumulated in previous updating steps, pushes the weights update so that the updating process can rollover the zero weight gradient area, and appeared in the latter advanced optimizing function.

$$v_t = (1-\gamma)g_t + \gamma\nabla L_{batch}(w)_{ij} \tag{2.13}$$

$$w'_{ij} = w_{ij} - \eta v_{new} \tag{2.14}$$

where g_t is the momentum function and γ is the hyper-parameter satisfying $0 \leq \gamma \leq 1$.

4. Adaptive learning rate

Adagrad is the optimizer with an adaptive learning rate and has the mathematical expression below. The learning rate becomes smaller if the weight gradient gets small, which is more reasonable, as it considered the updating of the weights based on the current weight state.

$$s_t = s_{t-1} + g_t \odot g_t \tag{2.15}$$

$$w'_{ij} = w_{ij} - \frac{\eta}{\sqrt{s_t + \epsilon}} g_t \tag{2.16}$$

The ϵ is a dummy factor that prevents the dominator is zero and often be 10^{-6}, the s_0 will be initialized as zero, and the symbol $\odot$ means element multiplication.

RMSprop is the optimizer very similar to Adagrad. Since in Adagrad, the factor s_t will keep increase, the Aagrad optimizer will have inefficient learning in the later stage. Instead of keeping to add up, the RMSprop optimizer did nothing but balanced the term s_t between the previous and current state.

$$s_t = \gamma s_{t-1} + (1 - \gamma) g_t \odot g_t \tag{2.17}$$

$$w'_{ij} = w_{ij} - \frac{\eta}{\sqrt{s_t + \epsilon}} g_t \tag{2.18}$$

Where γ is the hyper-parameter satisfying $0 \leq \gamma_1 \leq 1$.

Further substitution done by Adadelta optimizer was replacing the constant learning rate η to the dynamic factor $\triangle x$ associated with weights:

$$s_t = \gamma_1 s_{t-1} + (1 - \gamma_1) g_t \odot g_t \tag{2.19}$$

$$w'_{ij} = w_{ij} - \sqrt{\frac{\triangle x_{t-1} + \epsilon}{s_t + \epsilon}} g_t \tag{2.20}$$

$$\triangle x_t = \gamma_2 \triangle x_{t-1} + (1 - \gamma_2) g_t \odot g_t \tag{2.21}$$

where: The factor ϵ aims to stabilize the calculation. In most cases is very small such as 10^{-6}.

The Adam optimizer kept the s_t and $\triangle x_t$. once the hyper-parameter $\gamma$is fixed such as 0.9 and both x and s have zero initialization, those two factors will have a small value in the beginning as the only contributing term will multiply by 0.1. In order to ease the effect, Adam optimizer has re-normalized the s_t and $\triangle x_t$:

$$\hat{s_t} = \frac{s_t}{1 - {\gamma_1}^t} \tag{2.22}$$

$$\hat{\triangle x_t} = \frac{\triangle x_t}{1 - {\gamma_2}^t} \tag{2.23}$$

$$w'_{ij} = w_{ij} - \frac{\eta \triangle \hat{x}_t}{\sqrt{\hat{s}_t + \epsilon}} g_t \tag{2.24}$$

2.2.2.3 The activation function

1. The sigmoid function

   The sigmoid function is the mathematical function that has an "S" shape. For example, it can be the logistic function, the hyperbolic function, the Arctangent function, etc. It appears to be the first generation of activation function in deep learning. The S shape endows the sigmoid function relatively high speed of changing in the middle section comparing to the two polar extremes.

   - Logistic function

     Logistic function takes the formula $\sigma(x) = \frac{1}{1+e^{-x}}$ and output the value at [0,1], it has been pervasively used since it can concentrate the large real output to the interval. However, the shape of the sigmoid function has saturation at two polar extremes, which will slow the weight updating process down once the output after the layer gets small, then produces the vanishing gradient problem.

   - Hyperbolic tangent
     The hyperbolic tangent function has a very high resemblance with the logistic function but confines output to [-1,1] and centralizes the result to 0. It takes form $f(x) = \frac{e^{2x}-1}{e^{2x}+1}$.

2. Rectified Linear Unit (ReLU)

   The rectified linear unit function can solve the vanishing gradient problem. Its activated output range is [0,1]. The mathematical form for ReLu activation has the form:

$$\sigma(x) = \max(0, x) \tag{2.25}$$

   A lot of research has pointed out that the ReLU function can improve training effectiveness. However, the network will not pronounce the negative output, which might cause training insufficiency as well.[21]

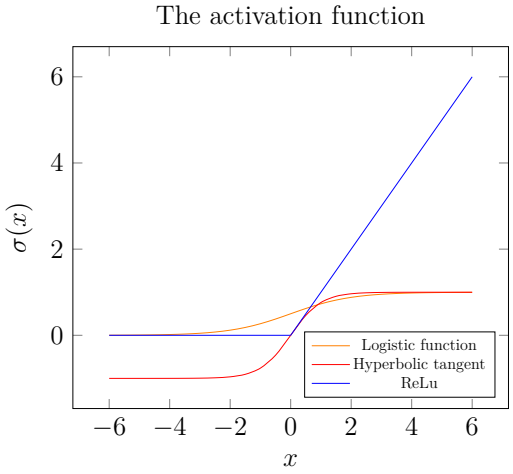


Figure 2.6: *The activation function*

2.3 Improve the network performance

2.3.1 Data Augmentation

Data augmentation is one data pre-processing technique that can enlarge the image database. Common was seen as flip, rotate, crop the original image. Data augmentation has proven benefits for improving performance. For example, it can generalize the network adaptability and prevent over-fitting.

2.3.2 Pre-processing

Data pre-processing includes re-scaling, standardization, normalization, etc. It can help the network learn effectively.

2.3.2.1 Re-scaling/Normalization

In general, the RGB image has 8 bits, which makes the pixel value range from 0 to 255. Large average and variation will decrease the training efficiency. Therefore, re-scaling the image to 0-1 is the first data prepossessing step for preparing the training data.

$$X = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.26}$$

Where X is the pixel value in image.
If one only applies the normalization on the original image, since $X_{max} = 255$ and $X_{min} = 0$. Therefore, the formula has the form:

$$X = X/255.0 \tag{2.27}$$

2.3.2.2 Standardization/Image whiting

The standardization process makes the standard deviation to 1 and the means value to 0, also known as the image whiting technique.

$$X = \frac{X - \mu}{\sigma} \tag{2.28}$$

2.3.3 Batch normalization

The batch normalization layer appears before non-linearity activation in general cases. The purpose of batch normalization is to accelerate the training process to reach the global minimum of loss. By adding the batch normalization layer, the mean value and standard variation can be 0 and 1, respectively.

The underpinning of the optimizing effect has many explanations. The most widely accepted one was reducing the internal con-variance shift, which means the negative impact of continuous weights updating proceeding between layers.[22] However, the study done by Shibani Santurkar et al.[23] has pointed out the positive effect attributed to the smoothing loss landscape during the training process mostly. The "smoothing effect" is favorable for the non-convex optimizing process by making the loss decreasing slowly with smaller weights and providing better "β-smoothness"[24]to the gradients of the loss. The author also pointed out, this smoothing effect does not bond to the batch normalization layer. It can get from other regularization methods too. [23]

2.3.4 Regularization

Regularization is the method used in deep learning to prevent over-fitting. It gives the modified loss function an additional penalty term(also be known as regularization term) so that the weight can be limited in a small range and leads to an uncomplicated model with a lower risk of over-fit training. The penalty term often has two type:
L1 regularization:

$$\widehat{L}(w) = L(w) + \frac{\alpha}{2} * \|w\| \tag{2.29}$$

$$\nabla_w \widehat{L}(w) = \alpha sign(w) + \nabla_w L(w) \tag{2.30}$$

$$w_{new} = w_{old} - \epsilon(\alpha sign(w_{old}) + \nabla_w L(w_{old})) \tag{2.31}$$

L2 regularization:

$$\widehat{L}(w) = L(w) + \frac{\alpha}{2} * \|w\|^2 \tag{2.32}$$

$$\nabla_w \widehat{L}(w) = \alpha w + \nabla_w L(w) \tag{2.33}$$

$$\begin{aligned} w_{new} &= w_{old} - \epsilon(\alpha w_{old} + \nabla_w L(w_{old})) \\ &= (1 - \epsilon\alpha) w_{old} - \epsilon \nabla_w L(w_{old}) \end{aligned} \tag{2.34}$$

Where α is the penalty factor and ϵ is the learning rate.

The L1 and L2 regularization subtract the absolute value of the weights matrix and the squared sum of the weight matrix, respectively. Therefore, L1 regularization can give a more sparse solution since the weights can be zero with a rate of -1 or 1.

2.3.5 Dropout

Dropout is a strategy for preventing over-fitting when the network becomes deep and complicated. It makes the network drop some neurons during each training iteration randomly with predefined probability p, which will force the remaining neurons to learn more effectively. However, one thing worth noticing is that due to the randomness of the turn-off neuron, it is unlikely to get the same inference prediction results.

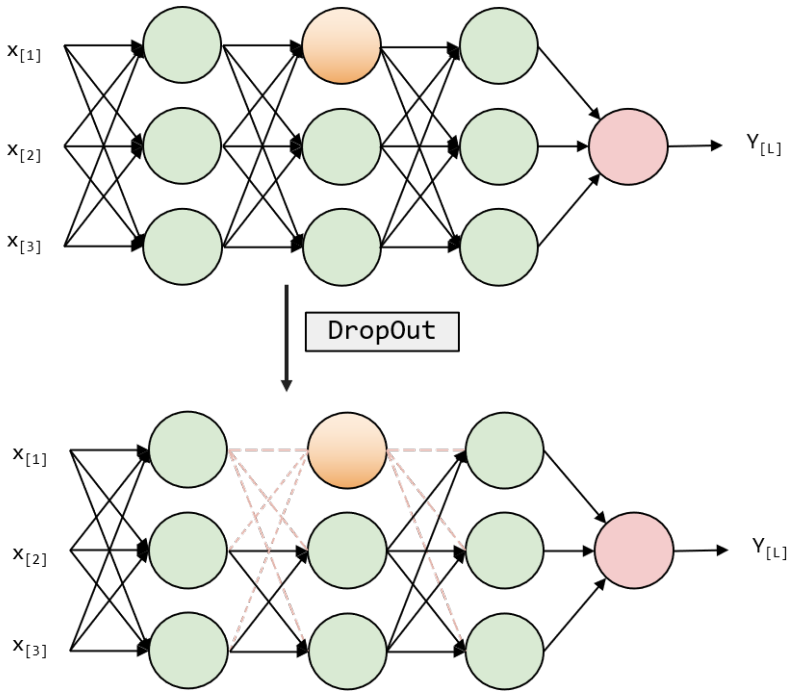


Figure 2.7: *Visualization of drop out*

3

Image Classification

This chapter aims at developing a proper convolutional neural network that can classify the input image accurately. In order to get the model with the best performance, multiple trials executed by altering different influential factors for the CNN will show up in the first section of this chapter. After this section, the best performance model will be compared with the transfer learning model which connects with different classifiers. Finally, the prediction results from both the transfer learning model and the fully-trained model are presented in results section.

3.1 Experiment Set-up

3.1.1 Image Data set preparation

The data-set used in this project is 77 camera-token RGB images. It contains various backgrounds including concrete scratches, stains, occlusion, wall edges, infrastructure ambient objects for example the grass and stone. The dimension for each image data is [1836,3264,3].

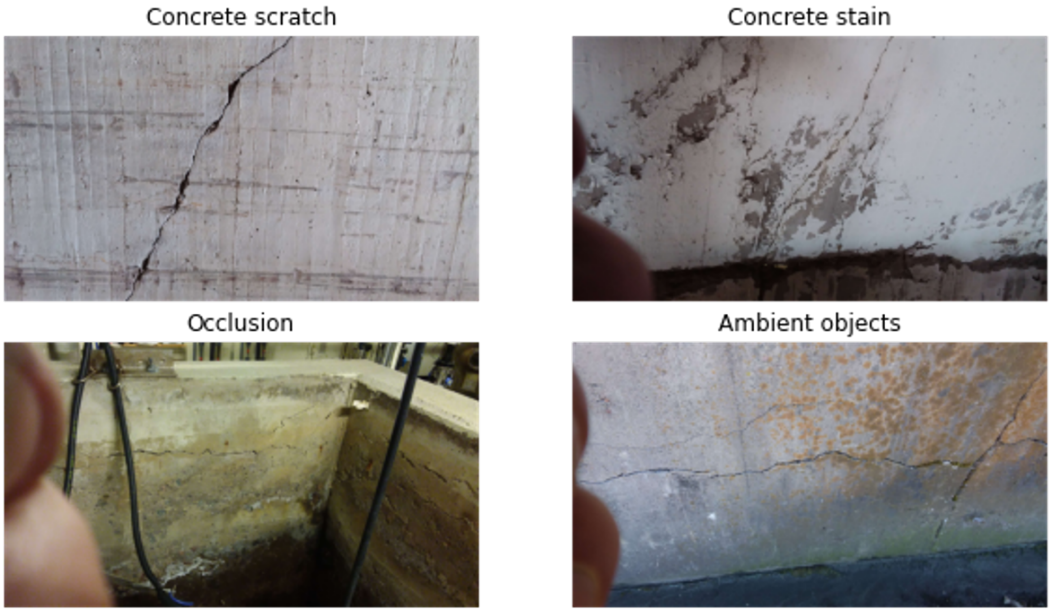


Figure 3.1: *Image example from database*

As an image processing technique, data augmentation can enlarge the database with limited image data, prevent over-fitting and favor the network generalization ability.

Rotating with a different angle, shearing, random brightness adjusting, and adding Gaussian noise has been applied in the experiment, chose which augmentation method was random.

Figure 3.2: *Data augmentation*

3.1.2 The convolutional neural network

The convolutional neural network used in the experiment has the architecture below:

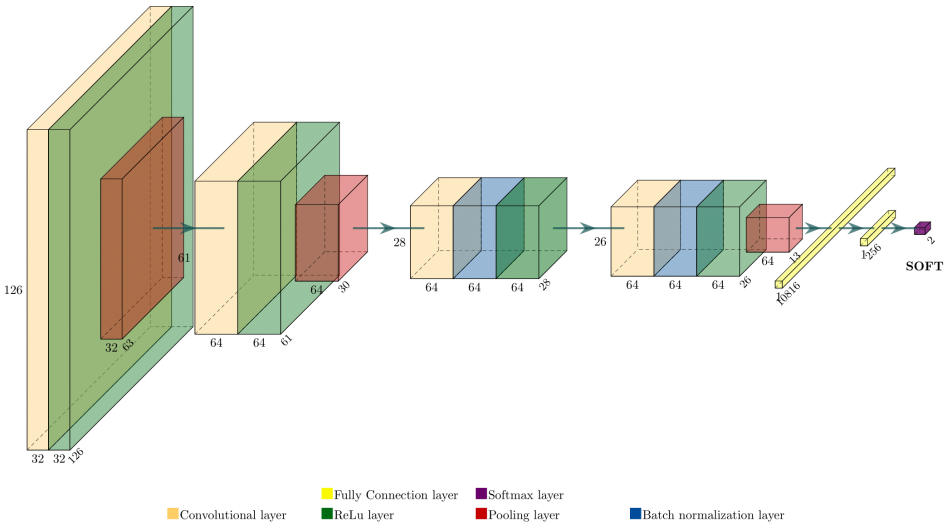


Figure 3.3: *The convolutional neural network*

3.1.3 Hardware and software environment

The models are training on Google Colab platform. The specification of the hardware and software environment is illustrated in Table 3.1

Table 3.1: *Specification of hardware and software environment*

Platform	Google colab
CPU	Intel(R) Xeon(R) CPU @ 2.20GHz
GPU	NVIDIA Tesla P100-PCIE (16GB)
Python	3.7.10
Packages	TensorFlow, 2.4.1; OpenCV2, 4.1.2; Numpy, 1.19.5; Matplotlib, 3.2.2

3.2 Implementation

3.2.1 Patch size

The sub-image size has a substantial effect on the network performance, as it will define how many pixels one image subject will contain. In this project, crack width generally propagates within the range of 8-16 pixels. But there also has a relatively high number of cracks that take more than 32 pixels. The sub-image size will alter the relative size between the crack feature and the un-cracked background, which might end up reflecting on the final classification results. Therefore, the investigation should be carried out about setting the appropriate sub-image size.

In order to carry out the single variable experiment, except for the patch size, which is 64, 128, and 256 respectively, the rest of the parameters will all be kept the same.

The total number of sub-images for each group is 8K, which contains half of the cracked images and the other half of the uncracked images. The train-validate-ratio keeps as 4:1, one-tenth of the total image was used for final evaluating.

Table 3.2: *The mutual hyper-parameter for training*

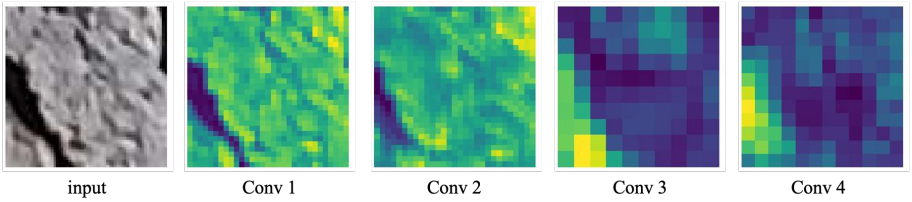
Optimizer	Learning rate	Epoch	Batch size
Adam	1e-05	1000	200

Table 3.3: *Comparison between different patch size*

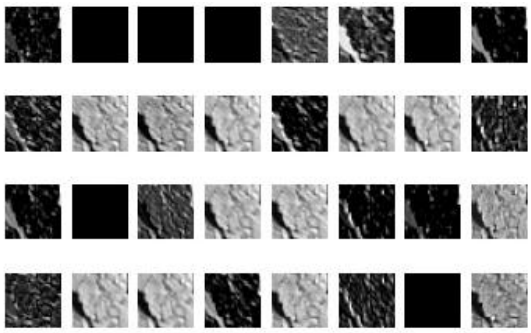
Patch size	Train acc(%)	Val acc(%)	Test acc(%)	Time(epoch/s)
64*64*3	89.81	84.51	83.75	1
128*128*3	90.74	84.24	84.62	2
256*256*3	96.88	92.50	89.63	7

As we can see from Table 3.3, the classification results are getting better when the patch size is increasing. In order to understand the results better, the feature map after each convolutional layer has been plotted for each patch size.

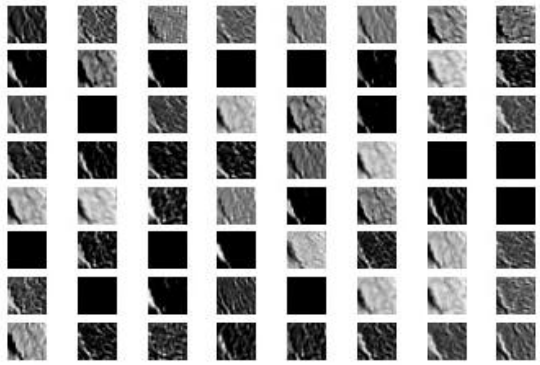
From Figure 3.4,3.5,3.6 below, we can see the first two convolutional layers can extract shallow features of the crack. When the convolutional layer gets deeper, a higher dimension of the features can show up after convolution. This phenomenon can explain why big patch size gives better prediction results in an intuitional way. A bigger patch size is favorable to the deep convolutional layers to extract more complex information about the crack features by providing a wider pixels range. On the other hand, a small patch size has no big scale of the information to give.



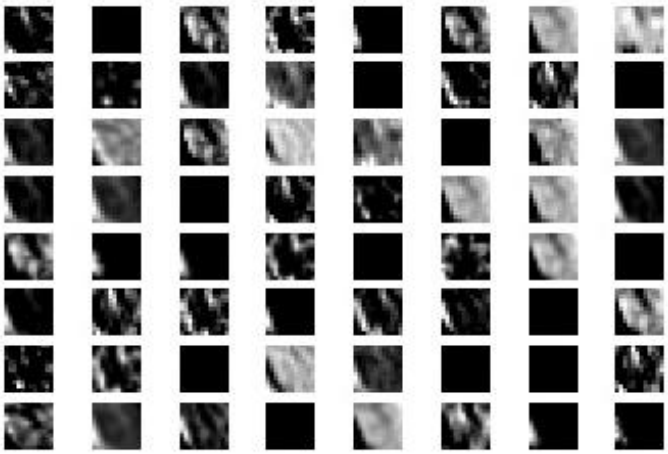
(a) Input image and the visualized output of each layer



(b) 1st Conv layer



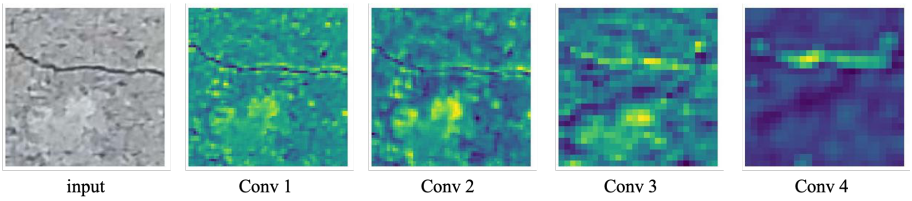
(c) 2nd Conv layer



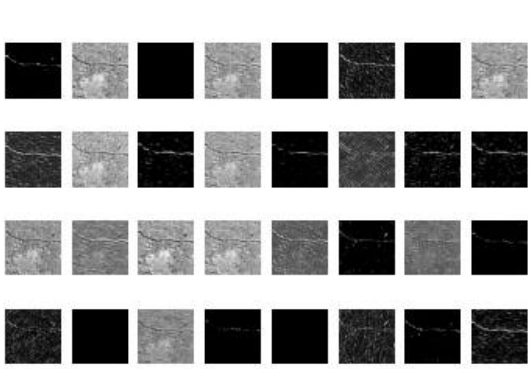
(d) 3rd Conv layer

(e) 4th Conv layer

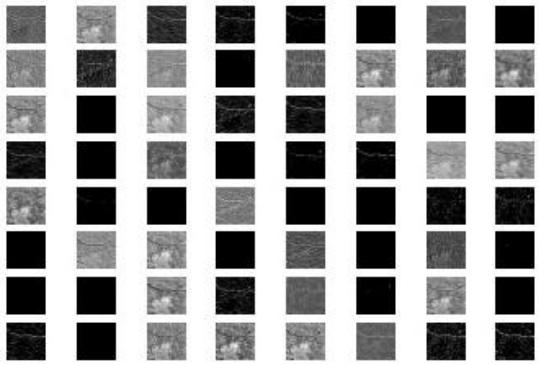
Figure 3.4: *The feature maps of each convolutional layer, patch size 64*64*



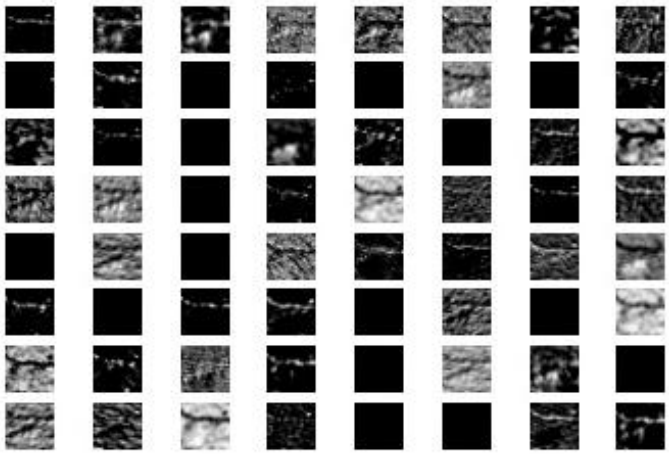
(a) *Input image and the visualized output of each layer*



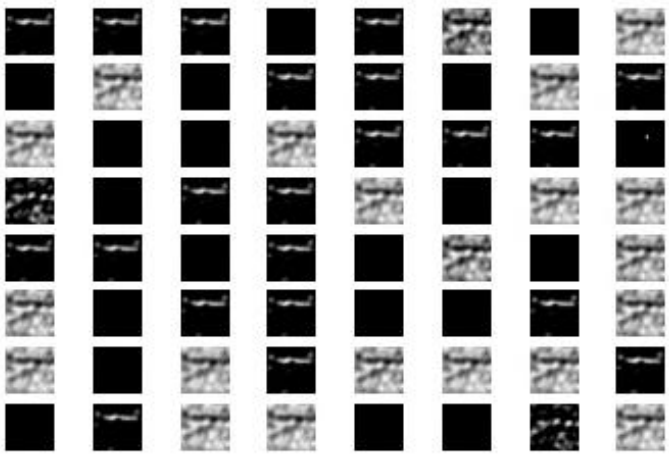
(b) *1st Conv layer*



(c) *2nd Conv layer*

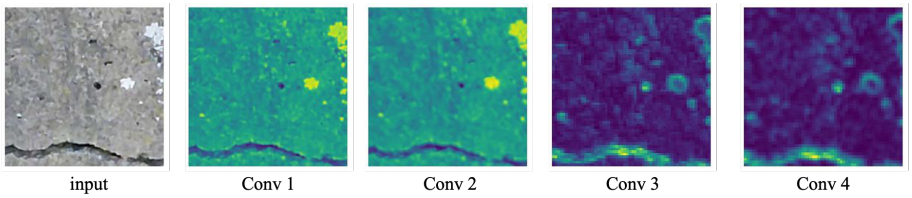


(d) *3rd Conv layer*

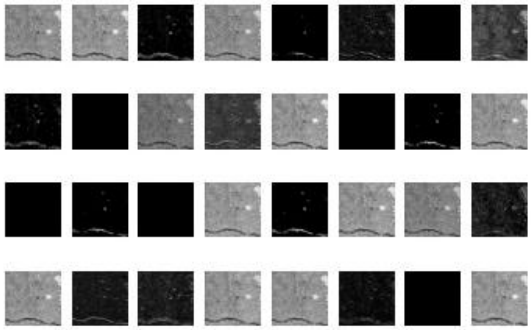


(e) *4th Conv layer*

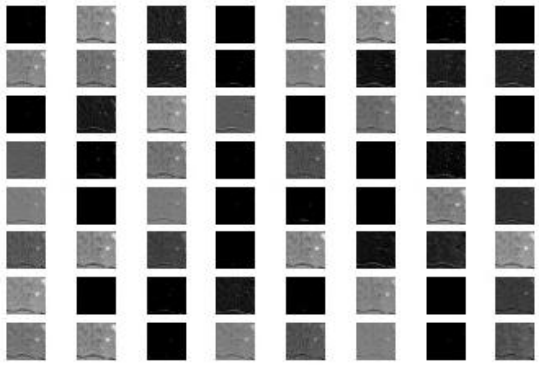
Figure 3.5: *The feature maps of each convolutional layer, patch size 128*128*



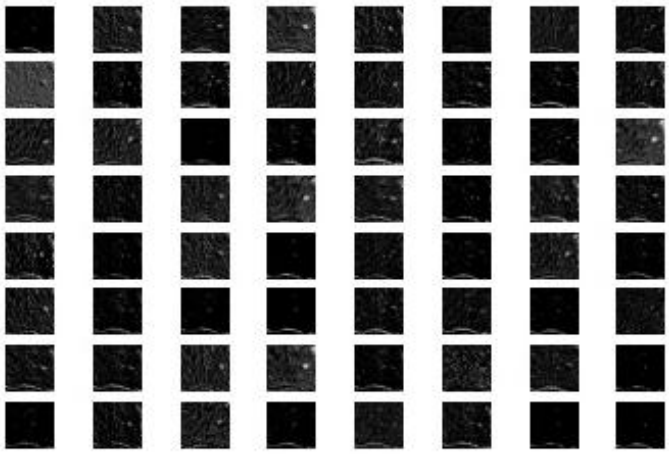
(a) *Input image and the visualized output of each layer*



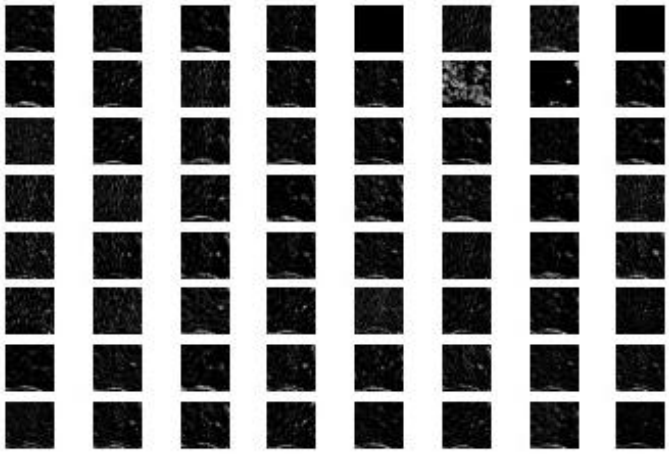
(b) *1st Conv layer*



(c) *2nd Conv layer*



(d) *3rd Conv layer*



(e) *4th Conv layer*

Figure 3.6: *The feature maps of each convolutional layer, patch size 256*256*

3.2.2 Parametric study

Database scale always is one of the most influential factors that can determine the final prediction results. There have a lot of studies investigated how big the influence of the database size can generate. In the parametric study, how the data constitution ratio and total training image number change the network performance will be discussed.

The scale study will compare the network performance under various total training image numbers. Except for the training image number, all the rest of the parameters will remain the same as Table 3.2 listed.

Table 3.4: *Comparison between different number of training images*

Image number	Train acc(%)	Val acc(%)	Test acc(%)	Time(epoch/s)
8000	91.50	89.00	89.05	2
10000	94.73	89.45	90.40	2
20000	94.03	92.11	92.15	4
30000	93.87	92.26	92.67	6
40000	93.86	93.28	93.52	8

The testing results suggest large database size has a positive effect on the training process. However, the most significant improvement is increasing the database from 10000 to 20000. Keep enlarging the training image number will improve the accuracy, but the training efficiency is getting down as the increasing rate gets slow. Considering the overall training efficiency, the total sub-image number for training will be 30K for the following experiment.

3.2.3 The architecture modification

3.2.3.1 The filter size

The original network has four convolutional layers with the same filter size. Based on the study did by Cha et al.,[25] the CNN architecture has several convolutional layers with different filter sizes, which performs exceptionally with sub-image dimension 256*256*3. For comparison purpose, network two modified with inspiration from the CNN built by Cha et al.[25] As Figure 3.7 shows, the modified network still has four convolutional layers. However, the convolutional filter size has replaced from universal 3*3 to 20*20,15*15,10*10 and 1*1 respectively. Moreover, the network has moved the batch normalization layer ahead and only uses the activation layer once. It is not hard to see the network uses a relatively wide stride size in the first three convolutional operations to reduce the dimension of the feature map.

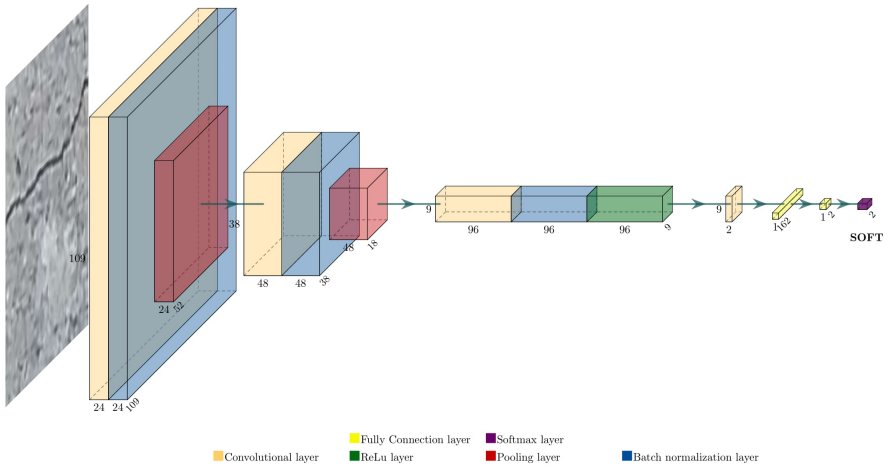


Figure 3.7: *The convolutional neural network with residue module*

Table 3.5: *Comparison between different network*

Network	Train acc(%)	Val acc(%)	Test acc(%)	Time(epoch/s)
Original	93.87	92.26	92.67	6
Modified	97.09	88.83	89.43	5

As Table 3.5 shows, the modified CNN has lower accuracy than the previous one. It is accredited to the stride size and potentially causes less convolutional layer.

3.2.3.2 Residue module

The residue module was first been proposed by Kaiming He, Xiangyu Zhang et al. in 2015.[26] Residue module is designed for training the deep neural network. The driven design motivation is predicated on the hypothesis that it is better for the network to find optimal solutions from the previous optimized layers than a stack of nonlinear layers.[26] By adding shortcuts for the deep layers, the identity layer from the previous layer will be utilized for helping the network find the optimal solution efficiently.

In this section, the strategy is to add the residue block to deepen the previous CNN architecture. The number of residue models is the single variable in the following experiment.

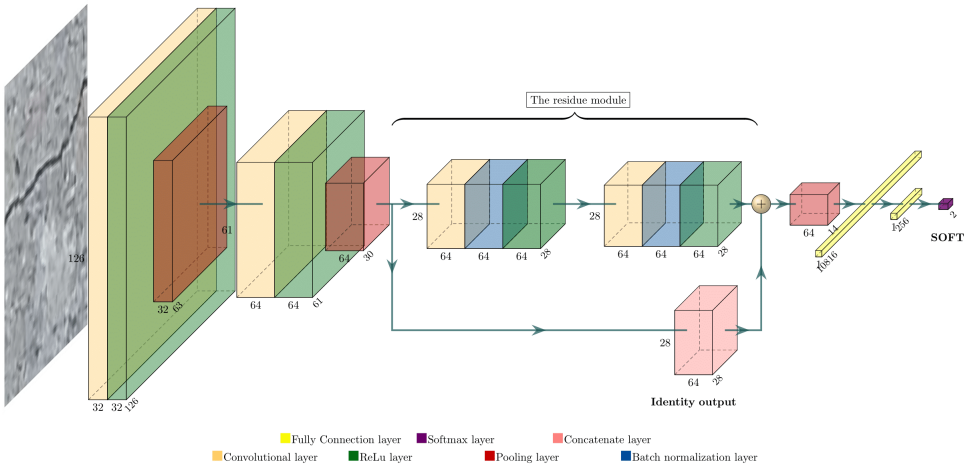


Figure 3.8: *The convolutional neural network with residue module*

Table 3.6: *Comparison between different residue module*

Num. Residue module	Train acc(%)	Val acc(%)	Test acc(%)	Time(epoch/s)
0	93.87	92.26	92.67	6
1	96.04	93.17	92.93	7
2	98.49	96.35	95.87	8
3	98.31	94.13	95.93	9

As we can see from Table 3.6 above, the network has the best performance when it has three residue blocks. But significant improvement hasn't been observed from increasing residue block two to residue block three, which suggests the limitation of gaining accuracy only by adding more residue block. The network architecture used in the rest of the experiment is the original network with three residue blocks.

3.2.4 The data imbalance

Data imbalance is a problem of data constitution when the data-set has a large unbalanced ratio for different data categories.

Since the un-cracked background is the dominating category of camera token photos in this project, the data augmentation technique was applied to ensure the cracked-uncracked ratio. Switching different train ratios can inevitably alter the classification results.

It is worth mentioning that when changing the cracked to uncracked ratio, the total image number will remain the same. Moreover, the testing data will change to five excluded full-scale images instead of the one-tenth image from the training dataset.

The motivation of changing testing data is to differentiate the data composition from training data to fully address the effect caused by the cracked and uncracked ratio. The model that has the highest validation accuracy during training will keep the same for the final test.

Table 3.7: *Comparison between different cracked and uncracked ratio*

Cracked-Uncracked ratio	Accuracy(%)	Recall(%)	Precision(%)	F1 Score
1:1	95.27	58.27	62.18	60.16
1:3	95.51	33.86	78.18	47.25
1:5	96.30	48.03	82.43	60.69
1:7	96.59	67.72	74.78	71.07

Table 3.7 suggests the best crack-uncracked ratio is 1:7. Although the results might be counter-intuitive at first, there has a non-negligible reason behind it. Since the total sub-image is unchanged as 30K, once the cracked image number decreased, the uncracked image number will inevitably increase. And the uncracked image in the training database has around 22K in total, which means if one increases the cracked parts, potentially cut-off some important uncracked feature input and decrease the accuracy in return. Another possible explanation is that when the uncracked portion increased, the training data constitution will approach the testing data. Therefore, the cracked-uncracked ratio for consisting training data is 1:7 for the rest of the experiment.

3.3 Transfer Learning

Transfer learning is a technique that utilizes the model trained on a large dataset to classify new data sets. However, the weights contained in the pre-trained model will not update since freezing pre-trained layers is the first step in the training process. One variation of transfer learning named fine-tuning chooses few layers not to be frozen but trained by the provided data. The transfer learning method is prevalent in the image classification field due to its convenience and capability of high accuracy.

3.3.1 The backbone of feature extractor

In this section, four pre-trained models will be the feature extractor, truncating the fully connected layers in the original model then replacing them with the classifier defined in the next section. The feature extraction means the convolutional process what done by multiple convolutional layers.

The pre-trained models are VGG16, Inception V3, MobileNet, and Densenet in Keras, which all include no classifier since the classifier will be custom-ed in the next section.

3.3.1.1 VGG16

VGG is the abbreviation of Visual Geometry Group, which designed this network to find how the depth of the network will affect the final network performance in 2014.[27] The noticeable difference with AlexNet is that VGG16 adopts the big convolutional filter size in AlexNet with continuous and unified size 3*3 filter size, which can reduce the training parameter for each layer to compensate for the increasing depth. This network design concept is also known as factorizing convolution. Table 3.8 below lists the VGG16 network architecture.

Table 3.8: *The network architecture of VGG16*

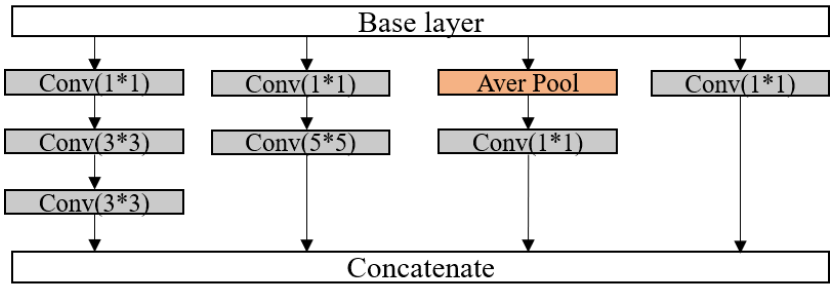
Layers	Output size	VGG16
Input	$128 \times 128 \times 3$	
Convolution	$128 \times 128 \times 64$	$3 \times 3 \times 64$ conv
Convolution	$128 \times 128 \times 64$	$3 \times 3 \times 64$ conv, stride 2
Max Pooling	$64 \times 64 \times 64$	$2 \times 2 \times 32$ max pooling
Convolution	$64 \times 64 \times 64$	$3 \times 3 \times 64$ conv
Convolution	$64 \times 64 \times 128$	$3 \times 3 \times 128$ conv
Max Pooling	$32 \times 32 \times 128$	$2 \times 2 \times 128$ max pooling
Convolution	$32 \times 32 \times 256$	$3 \times 3 \times 256$ conv
Convolution	$32 \times 32 \times 256$	$3 \times 3 \times 256$ conv
Convolution	$32 \times 32 \times 256$	$3 \times 3 \times 256$ conv
Max Pooling	$16 \times 16 \times 256$	$2 \times 2 \times 256$ max pooling
Convolution	$16 \times 16 \times 512$	$3 \times 3 \times 512$ conv
Convolution	$16 \times 16 \times 512$	$3 \times 3 \times 512$ conv
Convolution	$16 \times 16 \times 512$	$3 \times 3 \times 512$ conv
Max Pooling	$8 \times 8 \times 512$	$2 \times 2 \times 512$ max pooling
Convolution	$8 \times 8 \times 512$	$3 \times 3 \times 512$ conv
Convolution	$8 \times 8 \times 512$	$3 \times 3 \times 512$ conv
Convolution	$8 \times 8 \times 512$	$3 \times 3 \times 512$ conv
Max Pooling	$4 \times 4 \times 512$	$2 \times 2 \times 512$ max pooling
Classification layer	4096	$4 \times 4 \times 512 \times 4096$ fully connected
	1000	4096×1000 fully connected
		sotmax

3.3.1.2 Inception V3

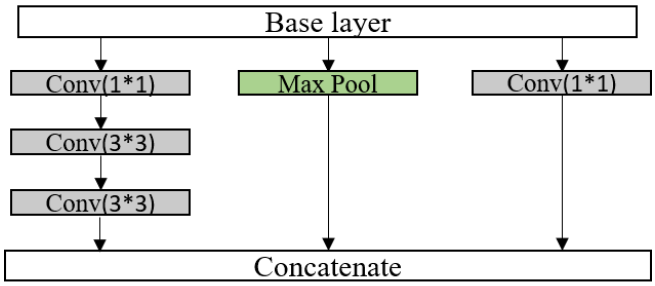
The inception V3 model has five different convolution module, which is all contains multi-scale convolution process. The difference between Inception version three and the previous version is that the added auxiliary classification part was just for regularization purposes.

Table 3.9: *The network architecture of Inception V3*

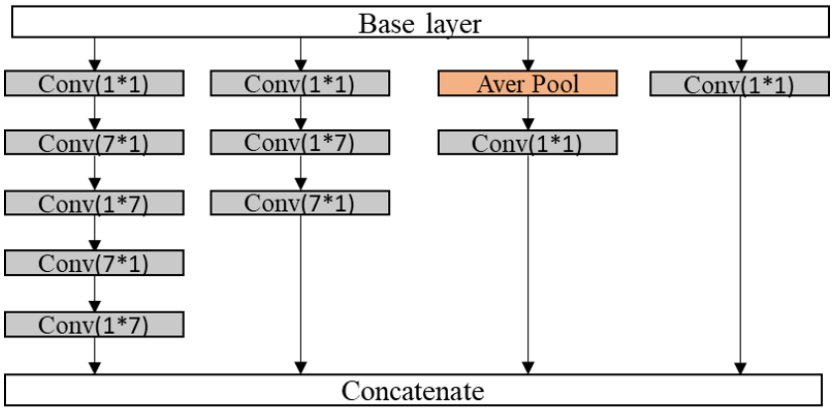
Layers	Output size	Inception V3
Input	$128 \times 128 \times 3$	
Convolution	$63 \times 63 \times 32$	$3 \times 3 \times 3 \times 32$ conv
Convolution	$61 \times 61 \times 32$	$3 \times 3 \times 32 \times 32$ conv
Convolution	$61 \times 61 \times 64$	$3 \times 3 \times 32 \times 64$ conv
Max Pooling	$30 \times 30 \times 64$	$2 \times 2 \times 64 \times 64$ conv
Convolution	$30 \times 30 \times 80$	$1 \times 1 \times 64 \times 80$ conv
Convolution	$28 \times 28 \times 192$	$3 \times 3 \times 80 \times 192$ conv
Max Pooling	$13 \times 13 \times 192$	$2 \times 2 \times 192 \times 192$ max pooling
Inception A $\times 3$	$13 \times 13 \times 288$	module processing
Inception B $\times 1$	$6 \times 6 \times 768$	module processing
Inception C $\times 4$	$6 \times 6 \times 768$	module processing
Inception D $\times 1$	$2 \times 2 \times 1280$	module processing
Inception E $\times 2$	$2 \times 2 \times 2048$	module processing
Average Pooling	2048	$2 \times 2 \times 2048$ global average pooling
Classification layer	1000	2048×1000 fully connected
		softmax



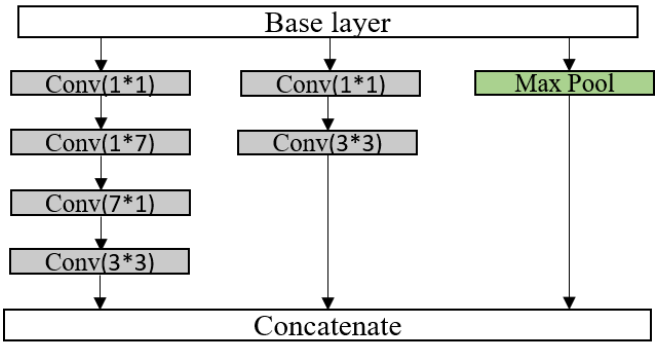
(a) *Inception module A layer*



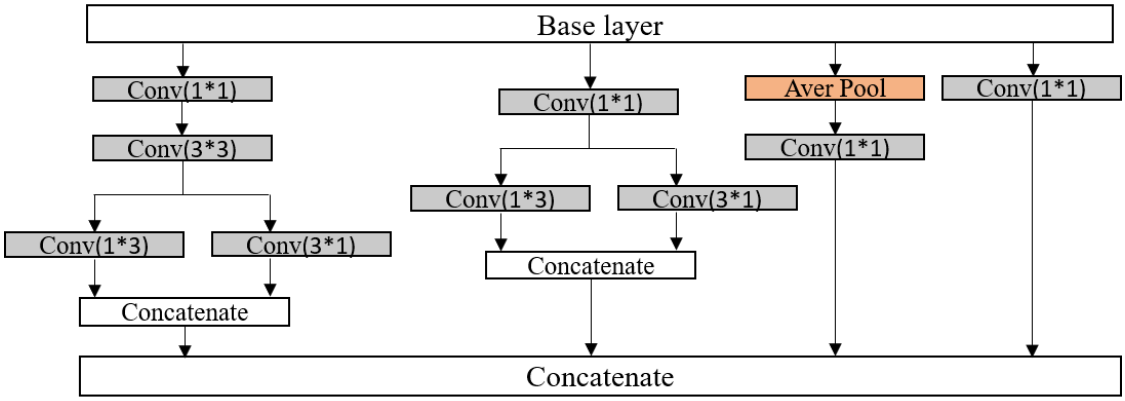
(b) *Inception module B*



(a) Inception module C



(b) Inception module D



(c) Inception module E

Figure 3.9: *Different modules in Inception V3 network*

3.3.1.3 MobileNet

Coined by Howard and Andrew G et al. at 2017[28], the MobileNet model applying the depth-wise separable convolution in the network design and has 28 layers in total. The so-called "depth-wise separable convolution work" includes two individual convolutional operations with different purposes. And the work is achieved by the depth-wise and point-wise convolutional layers. The former can filter the input

channel firstly then the latter can fuse the information. This elegant convolution design can significantly reduce the number of parameters needed in the traditional convolutional network.

Table 3.10: *The network architecture of MobileNet*

Layers	Output size	MobileNet
Input	$128 \times 128 \times 3$	
Convolution	$64 \times 64 \times 32$	$3 \times 3 \times 3 \times 32$ conv, stride 2
Convolution dw	$64 \times 64 \times 32$	$3 \times 3 \times 32$ conv, stride 1
Convolution pw	$64 \times 64 \times 64$	$1 \times 1 \times 32 \times 64$ conv, stride 1
Convolution dw	$32 \times 32 \times 64$	$3 \times 3 \times 64$ conv, stride 2
Convolution pw	$32 \times 32 \times 128$	$1 \times 1 \times 64 \times 128$ conv, stride 1
Convolution dw	$32 \times 32 \times 128$	$3 \times 3 \times 128$ conv, stride 1
Convolution pw	$32 \times 32 \times 128$	$1 \times 1 \times 128 \times 128$ conv, stride 1
Convolution dw	$16 \times 16 \times 128$	$3 \times 3 \times 128$ conv, stride 2
Convolution pw	$16 \times 16 \times 256$	$1 \times 1 \times 128 \times 256$ conv, stride 1
Convolution dw	$16 \times 16 \times 256$	$3 \times 3 \times 256$ conv, stride 1
Convolution pw	$16 \times 16 \times 256$	$1 \times 1 \times 256 \times 256$ conv, stride 1
Convolution dw	$8 \times 8 \times 256$	$3 \times 3 \times 256$ conv, stride 2
Convolution pw	$8 \times 8 \times 512$	$1 \times 1 \times 256 \times 512$ conv, stride 1
$5\times$ Convolution dw	$8 \times 8 \times 512$	$3 \times 3 \times 512$ conv, stride 1
$5\times$ Convolution pw	$8 \times 8 \times 512$	$1 \times 1 \times 512 \times 512$ conv, stride 1
Convolution dw	$4 \times 4 \times 512$	$3 \times 3 \times 512$ conv, stride 2
Convolution pw	$4 \times 4 \times 1024$	$1 \times 1 \times 512 \times 1024$ conv, stride 1
Convolution dw	$4 \times 4 \times 1024$	$3 \times 3 \times 1024$ conv, stride 2
Convolution pw	$4 \times 4 \times 1024$	$1 \times 1 \times 1024 \times 1024$ conv, stride 1
Average Pooling	1024	$4 \times 4 \times 1024$ global average pooling, stride 1
Classification layer	1000	1024×1000 fully connected
		softmax

3.3.1.4 DenseNet169

Table 3.11: *The network architecture of DenseNet169*

Layers	Output size	DenseNet 161
Input	$128 \times 128 \times 3$	
Convolution	$64 \times 64 \times 64$	7×7 conv, stride 2
Pooling	$32 \times 32 \times 64$	3×3 maxpool, stride 2
DenseNet Block (1)	$32 \times 32 \times 256$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	$32 \times 32 \times 128$	1×1 conv
	$16 \times 16 \times 128$	2×2 average pool,stride 2
DenseNet Block (2)	$16 \times 16 \times 512$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	$16 \times 16 \times 256$	1×1 conv
	$8 \times 8 \times 256$	2×2 average pool,stride 2
DenseNet Block (3)	$8 \times 8 \times 1280$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$
Transition Layer (3)	$8 \times 8 \times 640$	1×1 conv
	$4 \times 4 \times 640$	2×2 average pool,stride 2
DenseNet Block (4)	$4 \times 4 \times 1664$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$
Classification layer	1664	4×4 global average pool
		1000 fully connected, softmax

Table 3.11 shows the full scale of the DenseNet169 based on the 128*128*3 input image. Since the DenseNet169 is the backbone for extracting the feature, therefore includes no fully dense layer in the original network. Adding which classifier will achieve the best network performance is the content of the next section.

3.3.2 The classifier

3.3.2.1 The fully connection layer

After the last layer from the pre-trained model, the multi-dimensional data will be flattened first, then connected with trainable weights and bias in the dense layer. The weight updates were undergoing mainly in this part. In this experiment, the first dense layer consists of 256 neurons was successively followed by the relu activation and drop-out layer. The second dense layer contains two neurons representing the probabilities for the target output 1 or 0 after the softmax activation.

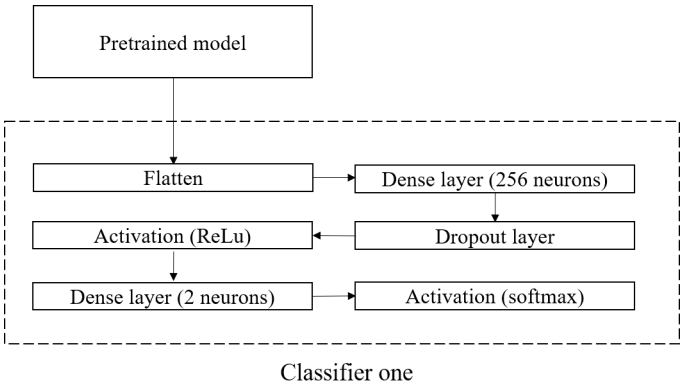


Figure 3.10: *The fully connected classifier*

Table 3.12: *Different feature extractor with fully connection layer*

Pretrained model	Accuracy(%)	Recall(%)	Precision(%)	F1 Score
VGG16	95.24	31.50	72.73	43.96
Inception V3	95.43	50.60	68.74	57.27
DenseNet169	96.74	62.20	78.22	69.30
MobileNet	97.11	66.14	81.55	73.04

3.3.2.2 Random forest

The random forest consists of many random decision trees. The building process of each tree was by picking data randomly first, then sub-divide the tree node based on the random feature in the data. The Figure below showcases one example of the growth of one random tree.

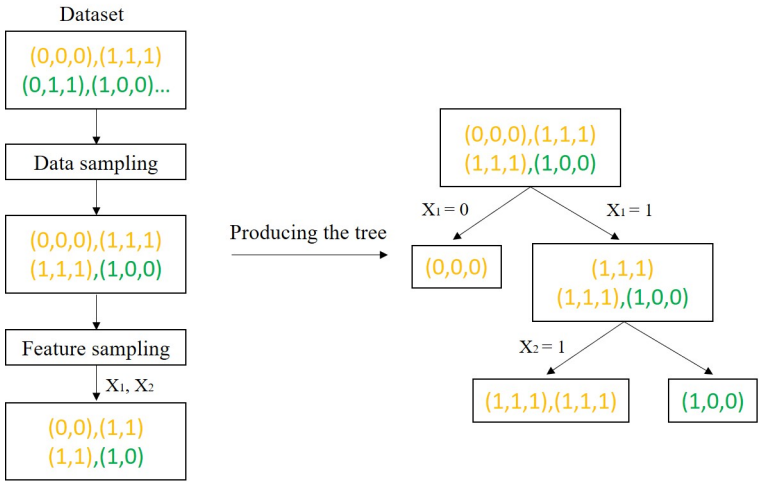


Figure 3.11: *The random forest classifier*

Table 3.13: *Different feature extractor with random forest classifier*

Pretrained model	Accuracy(%)	Recall(%)	Precision(%)	F1 Score
VGG16	95.88	38.58	83.05	52.69
Inception V3	95.23	28.35	76.60	41.38
DenseNet169	96.16	40.16	89.47	55.43
MobelNet	96.77	53.54	87.18	66.34

As we can see from Table 3.13, the random forest classifier has a lower recall value than the fully connected dense layer but higher precision. In this project, there needs to prioritize the recall ratio as the classification task should be on the passive side. Hence, the fully connected layer is more suitable for the classification task. But if one values precision more, the random forest classifier can also be a good choice, which also takes less time to train.

3.4 Results

As the testing results in the previous section has revealed, the fully connected layer with MobileNet backbone can offer highest F1 score for both classifier and even higher than the shallow CNN proposed in the previous section. Moreover, as Table 3.14 showed, the parameter for fully training the MobileNet is not far more than the shallow CNN. Therefore, it is worthy to implement the train-from-scratch process on MobileNet with more testing data. Although the accuracy is very high, the recall ratio still very low in all the training cases(the highest recall ratio was 67%). Since the recall parameter needs extra attention in the study, applying the focal loss function with parameter alpha as ten and gamma as one can ease the low recall problem. Test images were added from 6 to 15 to prevent over-fitting further. The total training image number still fixed as 3K. The cracked data up-sampling still applied. However, the ratio was not 1:7 anymore but filled by the augmented-cracked image and all the uncracked image.

The cost-sensitive loss function was a revised version of the binary cross-entropy function taking the form:

$$WeightedCrossEntropy = -w_0 y \log(p) - w_1(1-y)\log(1-p) \tag{3.1}$$

Where the w_o and w_1 are used to prioritizing specified category.

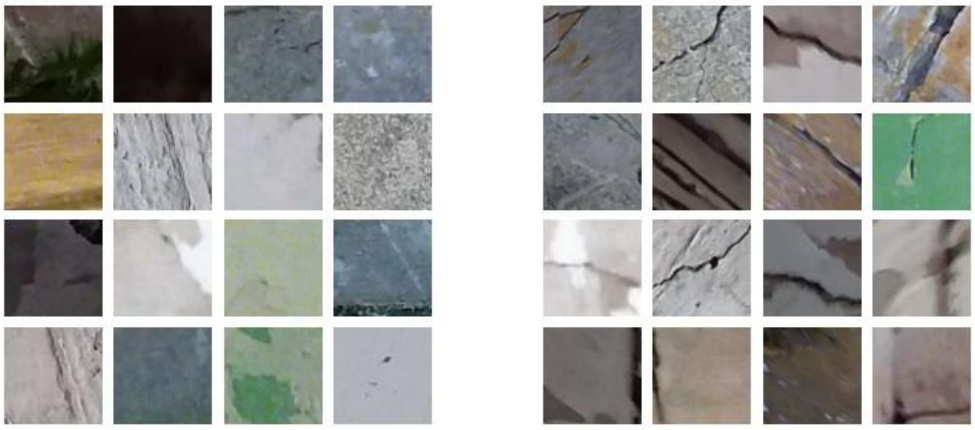
Table 3.14: *The parameter comparison*

Model	layers	Parameter(Million)
Shallow CNN	35	3.9
VGG16 transfered	26	16.8
Inception V3 transfered	318	23.9
DenseNet169 transfered	602	19.4
MobileNet transfered	93	7.4
MobileNet fully trained	94	4.2

Table 3.15: *Different network under cost sensitive loss function*

Pretrained model	Parameter (Million)	Accuracy(%)	Recall(%)	Precision(%)	F1 Score
MobileNet un-weighed	4.2	96.98	93.28	98.15	95.65
MobileNet $\alpha = 1$	4.2	97.29	94.70	97.54	96.10
MobileNet $\alpha = 0.75$	2.6	97.15	94.34	97.50	95.90
MobileNet $\alpha = 0.5$	1.3	96.91	94.34	96.79	95.55
MobileNet $\alpha = 0.25$	0.5	95.04	94.34	91.91	93.12
Shallow CNN	3.9	91.03	83.02	90.91	86.79

From Table 3.15, we can see as the alpha value decreases, the reduction on the recall ratio was not too much, but the precision ratio keeps dropping. When the alpha ratio decrease to 0.25, the accuracy has plunged to 95%. As mentioned previously, a high recall ratio suggests safe prediction is in this project. Al-through the parameter will increase when $\alpha = 1$, it will still be the final choice.

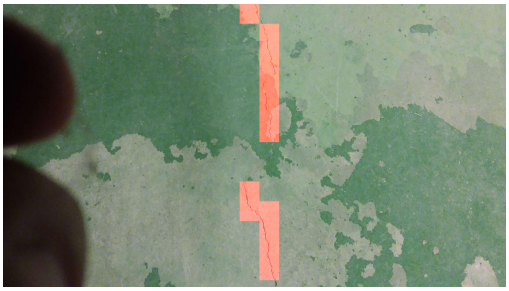


(a) *Uncracked image* ***(b)*** *Cracked image*

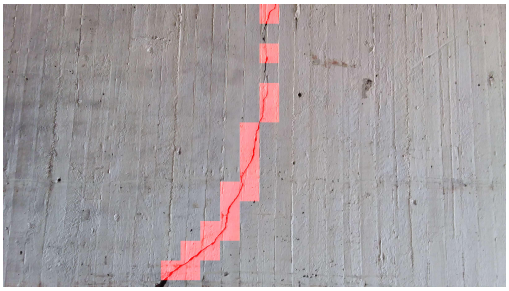
Figure 3.12: *The classification results*

The Figure 3.12 below shows an example coming out from the classification results of a fully trained Mobile network.

Based on the classification performance, it is reasonable to divide the testing image into three categories. They were the image has an ordinary concrete surface, contains noise, and has crack-like features. The network identifies the red patches as a cracked sub-image.

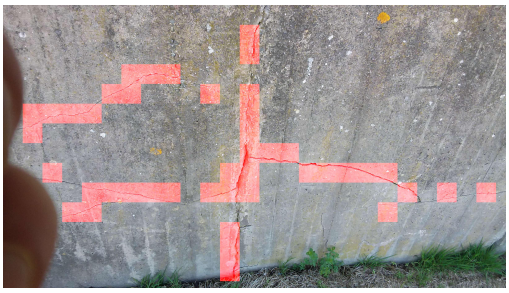


(a) *ordinary concrete surface image*



(b) *ordinary concrete surface*

(c) *Crack with noise image*



(d) *crack with crack-like features*

Figure 3.13: *The classification results*

From Figure 3.13, the whole scale picture for ordinary concrete surface has very accurate classification results. Some cracked sub-images are still missing in the sub-picture (d) and (c). It is reasonable to conclude that the network will be confused once one image has too many cracked-like features, then the recall ratio will decrease in consequence.

3.5 Conclusion

In this section, we have reached high accuracy in the cracked image classification task. A series of testing has run on choosing proper training parameters and different convolutional neural network architecture. The results show that the MobileNet model has very high training efficiency and exceptional performance on image classification. Using focal loss function and data up-sampling technique has to get a higher recall ratio. However, the results also show that if further optimization is needed, how to distinguish the crack-liked features and the authentic crack should be ameliorated. In this case, an image that contains too many crack-like features will impair the recall ratio, the Despite the limitation, the results still show that by providing the MobileNet appropriate training hyper-parameter, one can accomplish the crack classification task with high accuracy successfully.

4

Image Segmentation

Image segmentation refers to the process of recognizing and localizing different elements in a digital image. More specifically, different labels are assigned to each pixel in a digital image in the way that the pixels having the same label represent the same object.

In Chapter 3, a CNN classifier is developed which can recognize the presence of a crack in a sub-image context. But the sub-image resolution is still not sufficient for the subsequent structure performance prediction procedure. Though cracks have been located within the range of a sub-image, some information like the shape and the width is still missing, and these features of cracks can be essential for structure analysis.

The goal of this chapter is to develop an image segmentation technique that can extract cracks from a sub-image classified as positive by the previous CNN classification step. With the image segmentation processing, geometry information of cracks can be extracted and the resolution of the cracks will be increased from a sub-image patch to pixel level.

There are few reasons for performing image segmentation on sub-images rather than full-images. Full images usually cover a large area of the concrete surface and cracks are usually very small objects in a full-sized photo. For image segmentation based on CNN, the identifications of small objects are much more difficult to achieve, and area that is not related to a crack can become unnecessary interference, which not only consumes extra computational time, but is also harmful for the segmentation results. With the help of CNN classification, only the sub-images that related to cracks are concerned and all the other nun-concerned area is excluded, which will significantly increase the efficiency of the image classification process.

In this chapter, CNN models based on the famous U-net architecture for image segmentation tasks are established and trained with manually annotated crack sub-images. To address the issue of small object detection, different loss functions are used in the training. The models are tested on different kinds of crack sub-images and evaluated quantitatively and qualitatively. Finally, the best model is chosen considering both segmentation performance and computational efficiency.

4.1 U-net architecture

Convolutional neural networks are mostly used for data classification purpose. Image segmentation is basically the classification for pixels. This requires a network to understand the entire context of the input image. The U-net architecture is one of the most famous CNN architecture, which was first established for biomedical image processing and won Cell Tracking Challenge at ISBI in 2015 [29]. The architecture of U-net is shown in Figure 4.1.

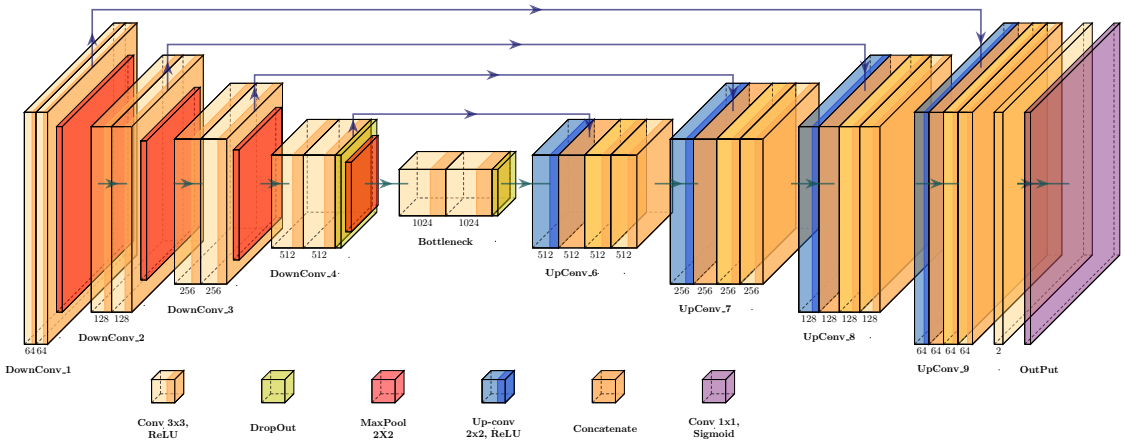


Figure 4.1: *U-net Architecture*

The U-net architecture has a symmetrical structure, which is similar to a encoder-decoder neural network. It consists of a down-sampling path, an up-sampling path and the skip connections between the layers of two paths. The interface of two paths is the 'Bottle Neck'. For both down-sampling and up-sampling paths, 4 convolutional units are employed. Table 4.1 shows the detail layer components of each unit. The stride value for all the convolutional kernels is 1 pixel on both direction, which means the kernel slides 1 pixel at a time on the input tensor. And the padding is set to 'same', which means the output feature maps of convolutional layers has the same size with the input tensor. The size of all the MaxPolling kernels is 2×2 and the stride value is 2, which means the output of MaxPolling layers has half the size of the input tensor.

Table 4.1: *Layer Components of Each Unit in U-net*

Unit	Layers	Kernel Size	Number of Feature Maps
DownConv_1	Conv, ReLU	3×3	64
	Conv, ReLU	3×3	64
	MaxPooling	2×2	-
DownConv_2	Conv, ReLU	3×3	128
	Conv, ReLU	3×3	128
	MaxPooling	2×2	-
DownConv_3	Conv, ReLU	3×3	256
	Conv, ReLU	3×3	256
	MaxPooling	2×2	-
DownConv_4	Conv, ReLU	3×3	512
	Conv, ReLU	3×3	512
	DropOut	-	-
	MaxPooling	2×2	-
BottleNeck	Conv, ReLU	3×3	1024
	Conv, ReLU	3×3	1024
	DropOut	-	-
UpConv_6	UpConv, ReLu	2×2	512
	Concatenate	-	[512+512]
	Conv, ReLU	3×3	512
	Conv, ReLU	3×3	512
UpConv_7	UpConv, ReLu	2×2	256
	Concatenate	-	[256+256]
	Conv, ReLU	3×3	256
	Conv, ReLU	3×3	256
UpConv_8	UpConv, ReLu	2×2	128
	Concatenate	-	[128+128]
	Conv, ReLU	3×3	128
	Conv, ReLU	3×3	128
UpConv_9	UpConv, ReLu	2×2	64
	Concatenate	-	[64+64]
	Conv, ReLU	3×3	64
	Conv, ReLU	3×3	64
Output	Conv, ReLU	3×3	2
	Conv, Sigmoid	1×1	1

In the up-sampling units, the size of input tensor is first doubled in UpConv layers and than convolutional transformation is performed, with kernels sized 2×2. Figure 4.2 shows a example of the UpConv layer. The extra columns and rows at the right-bottom corner are added to fulfill the same padding condition and the pixels of these area have values of 0. In this way, the outputs of UpConv layers are doubled-sized. After UpConv layers, the doubled-sized feature maps are concatenated with the feature maps from the corresponding unit in the down-sampling path and the

merged feature maps are processed with 2 more convolution layers.

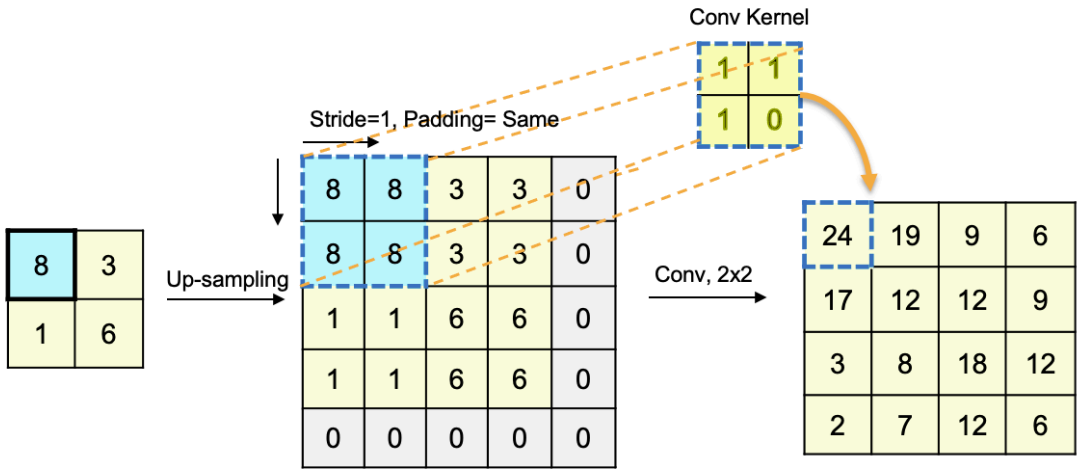


Figure 4.2: *An example of UpConv layer with 2×2 kernels, stride 1 and same padding*

The original U-net architecture is established for medical image segmentation, which is a much more complicated task than the identification of concrete cracks. It is not necessary to employ such a huge CNN model for crack detection purpose since it takes more computational power. In this study, smaller models are built in two ways. Firstly, the feature map numbers of each Conv layer are halved. Secondly, one unit is removed for both down-sampling and up-sampling path. In these ways. 4 architectures are established, namely Unet-4×64(original architecture), Unet-4×32, Unet-3 × 64 and Unet-3 × 32. The first number denotes the unit number in each path and the second number denotes the feature number of the first Conv layer in the down-sampling path. These models will be trained and compared and the model with the best performance will be chosen.

4.2 Data-set

Unlike CNN classification, the output of image segmentation CNN is not a single number that indicates sub-image categories, but a single channel image, where pixel values mean the possibility of being a crack. The ground truth of the training data should also be binary images, where a pixel can either be the background (0) or the crack (1).

230 sub-images sized 128 × 128 introduced in chapter 3 together with 220 images selected from METU data-set [30] are used to establish the data-set. The METU crack image data-set is a public image classification data-set. It provides images images sized 227 × 227 with binary labels indicating whether cracks present in the image but the pixel level annotation is not available. It is used here to enrich the data-set for training and among the METU images, 170 are resized to 128 × 128, and the rest will keep the size of 227 × 227 to generates more images by random

cropping.

The original crack images were annotated manually with 4 × 4 pixel patch, considering the balance of efficiency and accuracy. A binary image indicating the specific location and shapes of the cracks in the image is generated for each crack sub-image. Figure 4.3 shows few crack sub-images and the corresponding annotations.

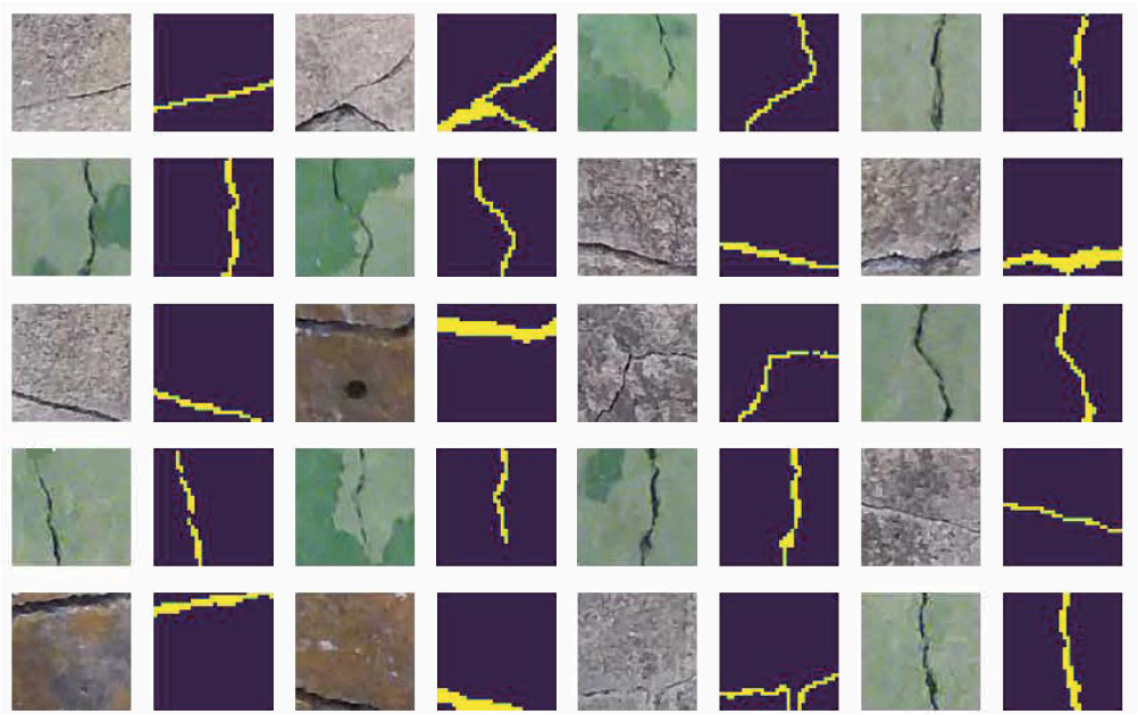


Figure 4.3: *Visualization of the Annotated Crack Image data-set*

The amount of crack sub-images is still far too less to train a neural network having millions of parameters. Data augmentation is applied to expand the data-set. For the sub-images sized 227 × 227, 4 images sized 128 × 128 are generated by random cropping. For all images sized 128 × 128 including those cropped images, 7 more images are further generated by flipping and rotating. Figure 4.4 and 4.5 show examples of the data augmentation for sub-images with different sizes.

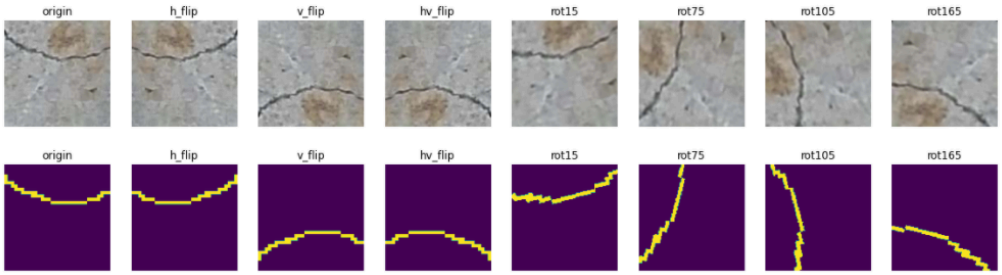


Figure 4.4: *An example of flip and rotation for images sized 128×128*

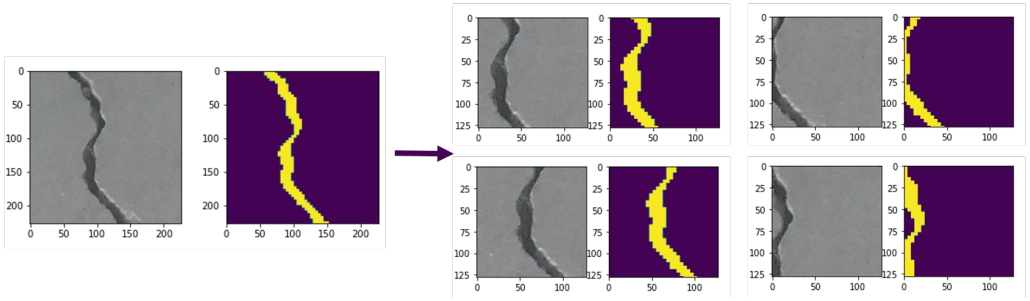


Figure 4.5: *An example of random cropping for images sized 227×227*

As a result, the final data-set has 4800 sub-images and annotation. The data-set is divided into training, validation and test sets according to a ration of 24:5:1, i.e, 3840 for training, 800 for validation and 160 for testing.

4.3 Unbalanced segmentation and loss functions

For the specified crack segmentation task, a major obstacle for training is the fact that cracks are usually very small objects in the sub-images. In the 4800 sub-images, positive pixels only take approximately 10%. This unbalance of two classes can causes problems for CNN models to recognize the target area in the input images. Specifically, false negative predictions creates incredibly less loss than false positives prediction during the training, thus the model learns more about the negative area than the positive area and results in a poor performance.

One of the reasons for this unbalanced segmentation problem is the fact that the commonly used loss function Binary Cross Entropy **(BCE)** treats the false positive and false negative predictions equally. Researches in this area have tried to change the definition of the loss function for the training. In this study, to generalize a suitable models for crack segmentation task, models trained with generalized loss functions are compared.

The definitions of generalized loss functions used in this study is introduced in this section. Without loss of generality, the formulations of the selected loss functions are expressed in multi-class classification case. For class c, float variable $p_{nc} \in [0, 1]$ represents the probability. Binary variable $g_{nc} \in \{0, 1\}$ is the ground truth of pixel n being that class. N is the total number of pixels in a sub-image. ϵ is a small real number to prevent the situation of dividing by zero.

1. **Binary Cross Entropy (BCE)**: As introduced in subsection 2.2.2.1, Binary Cross Entropy is the most common loss function. The formula of BCE can be written as:

$$BCE = -\sum_{n=1}^{N} g_{nc} \log(p_{nc}) + (1 - g_{nc}) \log(1 - p_{nc}) \tag{4.1}$$

2. **Dice Loss(DCL)**: Dice Loss is established based on the Dice Score Coefficient **(DSC)**, which describe the overlap extent of the predicted result and the ground truth.[31]. The formulation of Dice Score Coefficient is:

$$DSC = \sum_{c} \frac{\sum_{n=1}^{N} p_{nc} g_{nc} + \epsilon}{\sum_{n=1}^{N} p_{nc} + \sum_{n=1}^{N} g_{nc} + \epsilon} \tag{4.2}$$

Dice Loss **(DCL)** is a target function to be minimized in the training, first proposed by Milletari et al.[32]:

$$DCL = 1 - DSC \tag{4.3}$$

3. **Tversky Loss(TL)**: Tversky Index **(TI)** also describes the overlap of prediction and ground truth, but has different weights on false negative and false positive predictions by coefficients α and β [33]. When $\alpha = \beta = 0.5$, Tversky Index is equivalent to Dice Score Coefficient. In this study, the values for these coefficients are $\alpha = 0.7$ and $\beta = 0.3$, since for small target segmentation tasks false negative predictions have more influence on the final prediction. The expression for Tversky Index **(TI)** and Tversky Loss **(TL)** are:

$$TI = \sum_{c} \frac{\sum_{n=1}^{N} p_{nc} g_{nc} + \epsilon}{\sum_{n=1}^{N} p_{nc} g_{nc} + \alpha \sum_{n=1}^{N} (1 - p_{nc}) g_{nc} + \beta \sum_{n=1}^{N} p_{nc} (1 - g_{nc}) + \epsilon} \tag{4.4}$$

$$TL = 1 - TI \tag{4.5}$$

4. **Focal Tversky Loss(FTL)**: Abraham et al.[34] proposed Focal Tversky Loss by adding a exponent index γ to Tversky Loss. The index γ is smaller than 1 and in this way, when TI is getting large, which means a better performance, FTL decreases more significantly. In this study the γ is set to 0.75. The expression for FTL is:

$$FTL = (1 - TI)^{\gamma} \tag{4.6}$$

4.4 Evaluation Criteria

In this study, the evaluations of models are based on two indicators, Sensitivity **(SEN)** and Specificity **(SPEC)**. These tow indicators are the correction ratio for each class and represent the ability to recognize cracks and background respectively. Sensitivity and Specificity can be expressed by the number of true positive **(TP)**, true negative **(TN)**, false positive **(FP)** and false negative **(FN)** predictions, shown in equation (4.7).

$$SEN = \frac{TP}{TP + FN} SPEC = \frac{TN}{TN + FP} \tag{4.7}$$

Figure 4.6 shows an example of prediction results and corresponding sensitivity and specificity values.

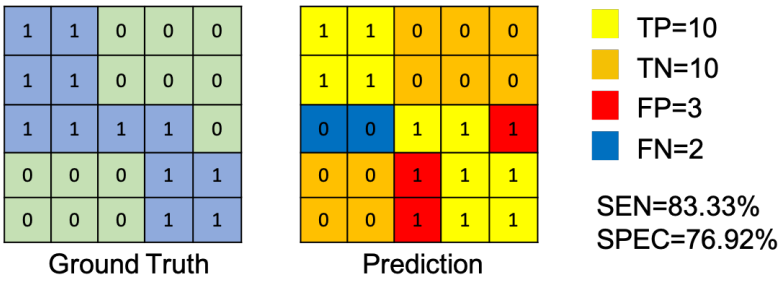
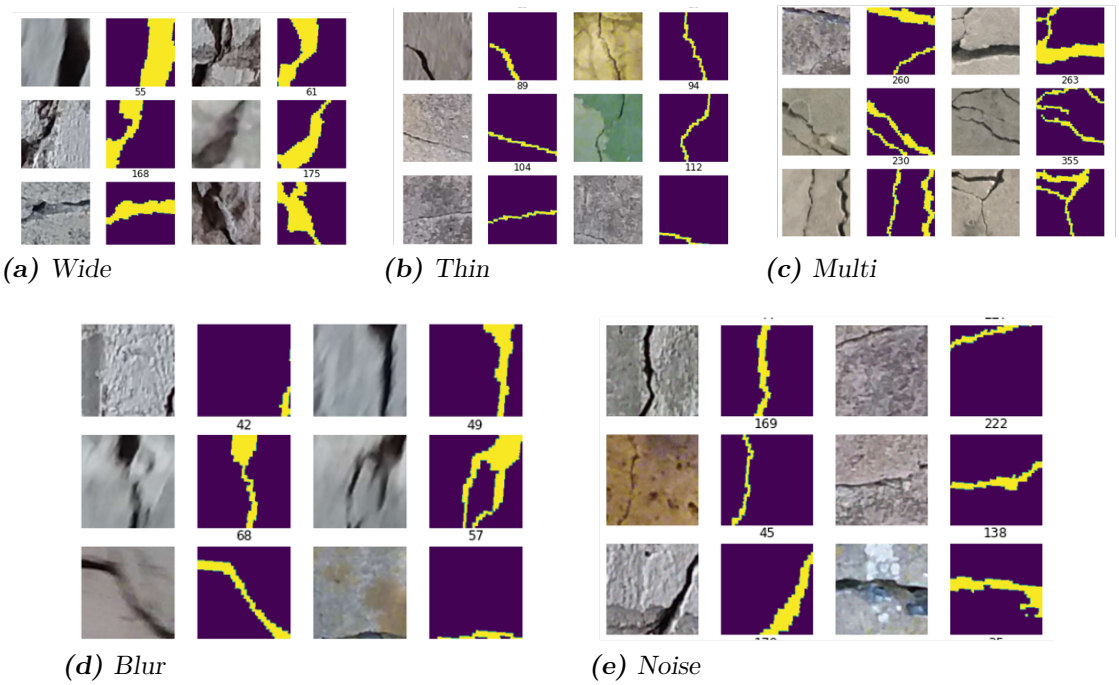


Figure 4.6: *An example of true positive, true negative, false positive and false negative predictions*

In the field application the model faces complicated situations of input images. To evaluate the performance of the models under different situations, the test data set are divided into five groups of wide, thin, multi, blur and noise (shown in Figure 4.7) manually and the model performance is evaluated on each group in terms of mean SEN and SPEC.



(a) *Wide* ***(b)*** *Thin* ***(c)*** *Multi*

(d) *Blur* ***(e)*** *Noise*

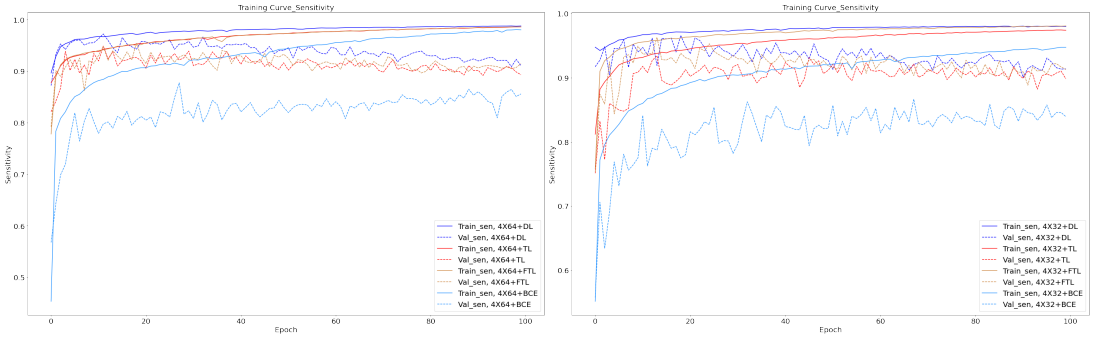
Figure 4.7: *Different groups of test images for evaluation*

Lastly, an important aspect of concern is the computational cost of each model, which is only related to the architecture of models but not influenced by the training methods or loss functions. These four architectures mentioned in section 4.1 will be tested on same computational environment and compared in terms of mean processing time for one sub-image.

4.5 Training and Results

The models are built and trained on Google Colab platform, the same machine learning environment used in Chapter 3. The specifications of hardware and software environment are listed in Table 3.1

All the models are trained for 100 epochs on the training data-set, with optimizer Adam and learning rate 10^{-4}. After each training epoch the Loss, SEN and SPEC value for training and validation set is recorded to plot training curves. Figure 4.8 shows the training curves for sensitivity value, for the same model trained with different loss functions. From the training curves it can be concluded that DCL, TL and FTL can all improve the training progress and models trained with these loss functions converge at a higher value of sensitivity, resulting in a batter performance than those trained with BCE. But the effects of these loss functions differ for different architectures.

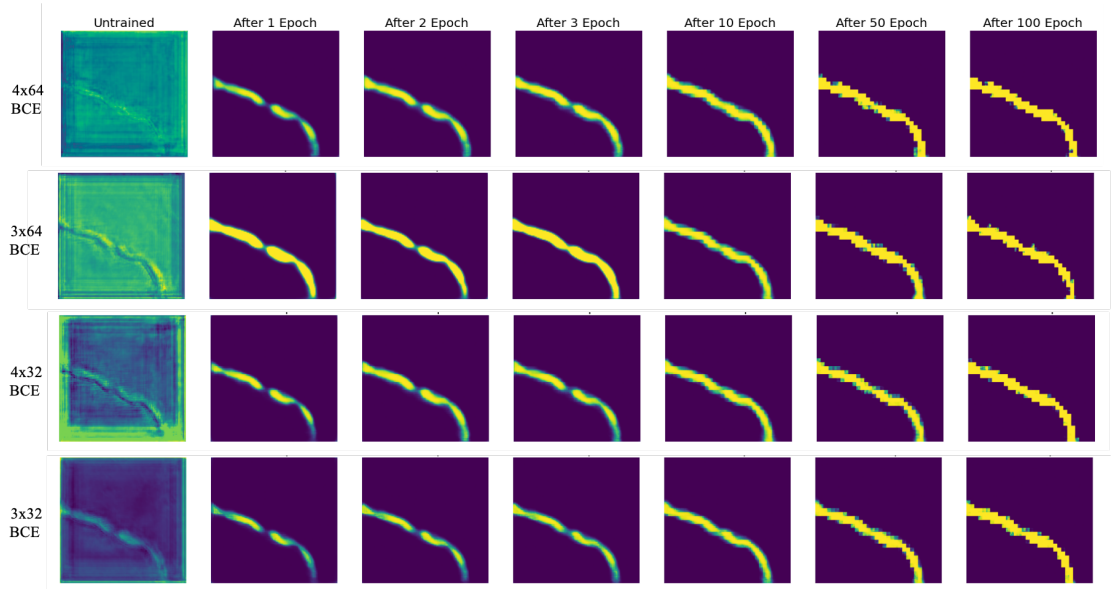


(a) *Model 4X64* ***(b)*** *Model 4X32*

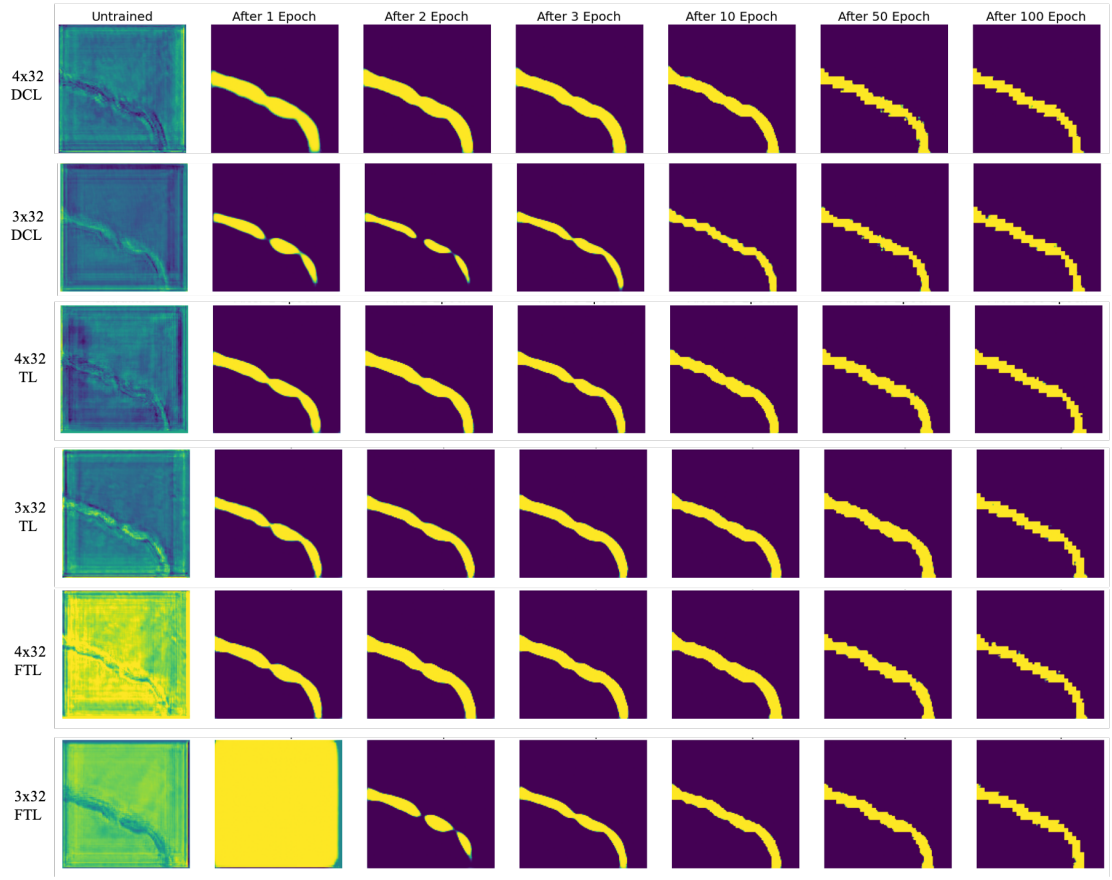
(c) *Model 3X64* ***(d)*** *Model 3X32*

Figure 4.8: *Training curves of models trained with different loss functions*

A prediction of an certain sub-image from validation set is made after each epoch to learn and compare how different models progress. Figure 4.9 shows the predictions of the chosen validation sub-image after first 3 epochs and after 10 and 100 epochs for each training. The first prediction is a result of the random initialization of the model parameters.



(a) Models trained with BCE



(b) Models trained with other loss functions

Figure 4.9: *Training progression of different models*

Table 4.2 and 4.3 shows the performance of different models in terms of Sensitivity and Specificity on different test image groups as introduced in section 4.4. The global column shows the indicator values for the whole test set. The optimal results for each column are marked with bold font.

Table 4.2: *Sensitivity of different models*

Loss Functions	Architecture	Sensitivity (%)					
		Thin	Wide	Multi	Noise	Blur	Global
BCE	Unet-4x64	73.79	90.04	74.80	72.47	76.18	82.64
	Unet-4x32	71.46	89.70	71.33	74.77	73.80	80.32
	Unet-3x64	76.73	90.00	76.07	71.99	76.23	82.72
	Unet-3x32	77.58	91.30	77.62	70.05	74.33	82.44
FTL	Unet-4x64	84.62	**94.89**	84.48	79.11	**87.24**	87.62
	Unet-4x64	83.47	94.62	83.88	77.07	84.24	86.49
	Unet-4x64	84.16	93.46	83.73	79.23	85.14	86.66
	Unet-4x64	81.18	91.31	81.44	71.74	77.24	82.82
TL	Unet-4x64	83.04	93.04	81.50	78.92	83.40	85.63
	Unet-4x32	79.78	93.65	81.88	74.98	84.55	84.68
	Unet-3x64	86.58	93.28	84.62	80.16	82.43	87.12
	Unet-3x32	82.03	93.95	83.19	76.78	81.24	85.36
DCL	Unet-4x64	85.99	94.49	83.64	**81.16**	86.24	87.85
	Unet-4x32	**88.35**	94.18	**85.07**	78.71	87.07	**88.36**
	Unet-3x64	74.59	89.31	76.49	72.35	75.38	79.56
	Unet-3x32	77.02	91.49	77.27	73.96	76.53	81.39

Table 4.3: *Specificity of different models*

Loss Functions	Architecture	Specificity (%)					
		Thin	Wide	Multi	Noise	Blur	Global
BCE	Unet-4x64	98.93	98.71	98.31	98.69	99.12	98.76
	Unet-4x32	99.03	99.00	98.35	98.83	99.24	**98.92**
	Unet-3x64	99.04	98.74	98.26	98.78	99.21	98.81
	Unet-3x32	**99.12**	**99.02**	**98.65**	97.72	**99.35**	98.86
FTL	Unet-4x64	98.44	97.86	97.14	97.94	98.41	97.99
	Unet-4x64	98.44	97.78	97.27	97.90	98.58	98.00
	Unet-4x64	98.39	97.93	97.02	98.08	98.57	98.02
	Unet-4x64	98.50	98.17	97.30	98.40	98.88	98.25
TL	Unet-4x64	98.55	98.33	97.48	98.44	98.85	98.34
	Unet-4x32	98.68	98.16	97.56	98.62	98.78	98.34
	Unet-3x64	98.22	97.89	97.00	98.36	98.79	98.02
	Unet-3x32	98.46	97.80	97.20	98.20	98.57	98.04
DCL	Unet-4x64	98.38	97.94	97.28	98.51	98.61	98.12
	Unet-4x32	98.18	97.86	97.10	98.34	98.54	97.98
	Unet-3x64	98.93	98.79	98.12	**98.85**	99.24	98.79
	Unet-3x32	98.85	98.58	98.01	98.38	99.04	98.60

The performance of different models are also compared visually. Figure 4.10 shows the prediction results of Unet-4x32 models trained with different loss functions. Figure 4.11 shows the prediction results of different model architectures trained with DCL.

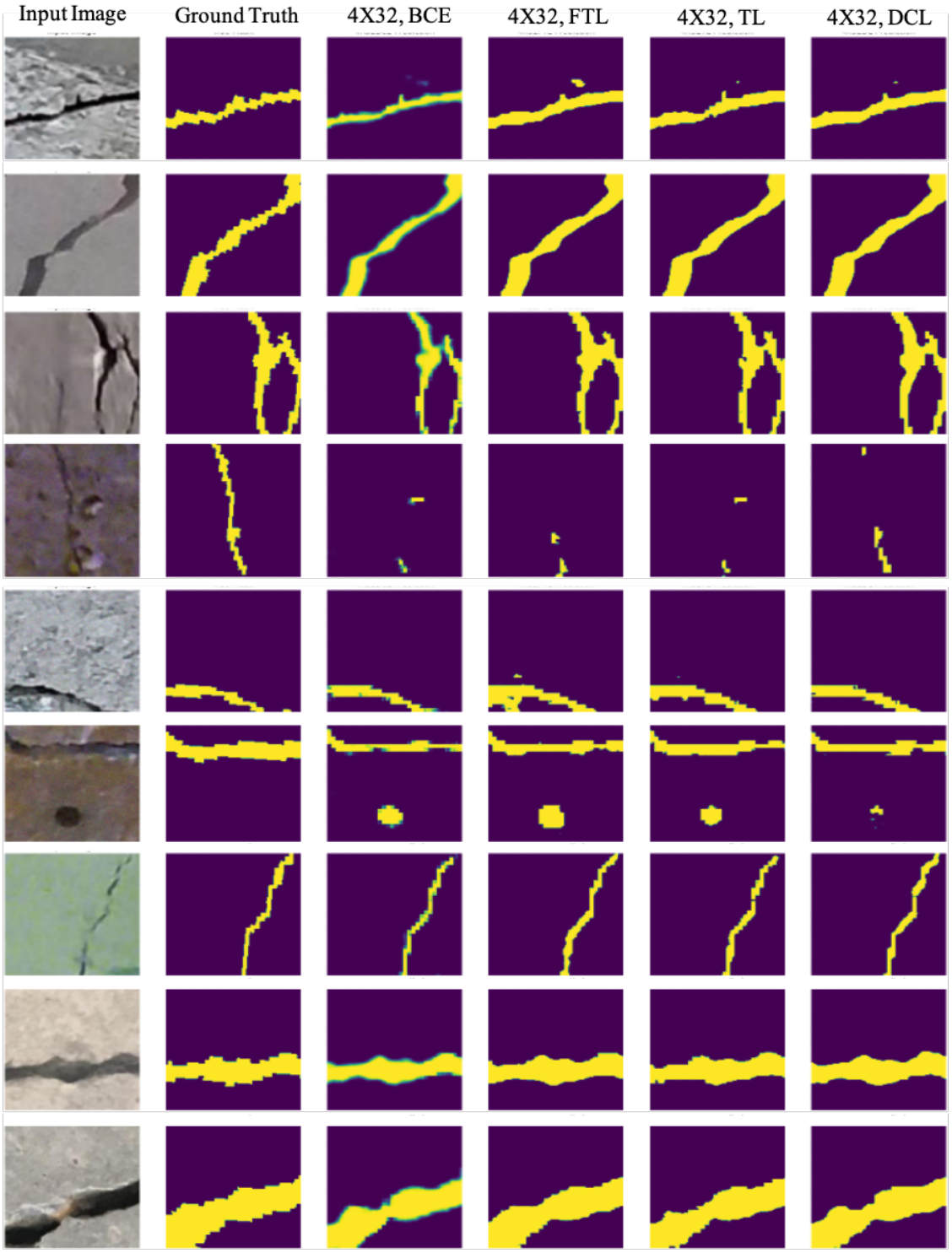


Figure 4.10: *Comparison of Unet-4x32 models trained with different loss functions*

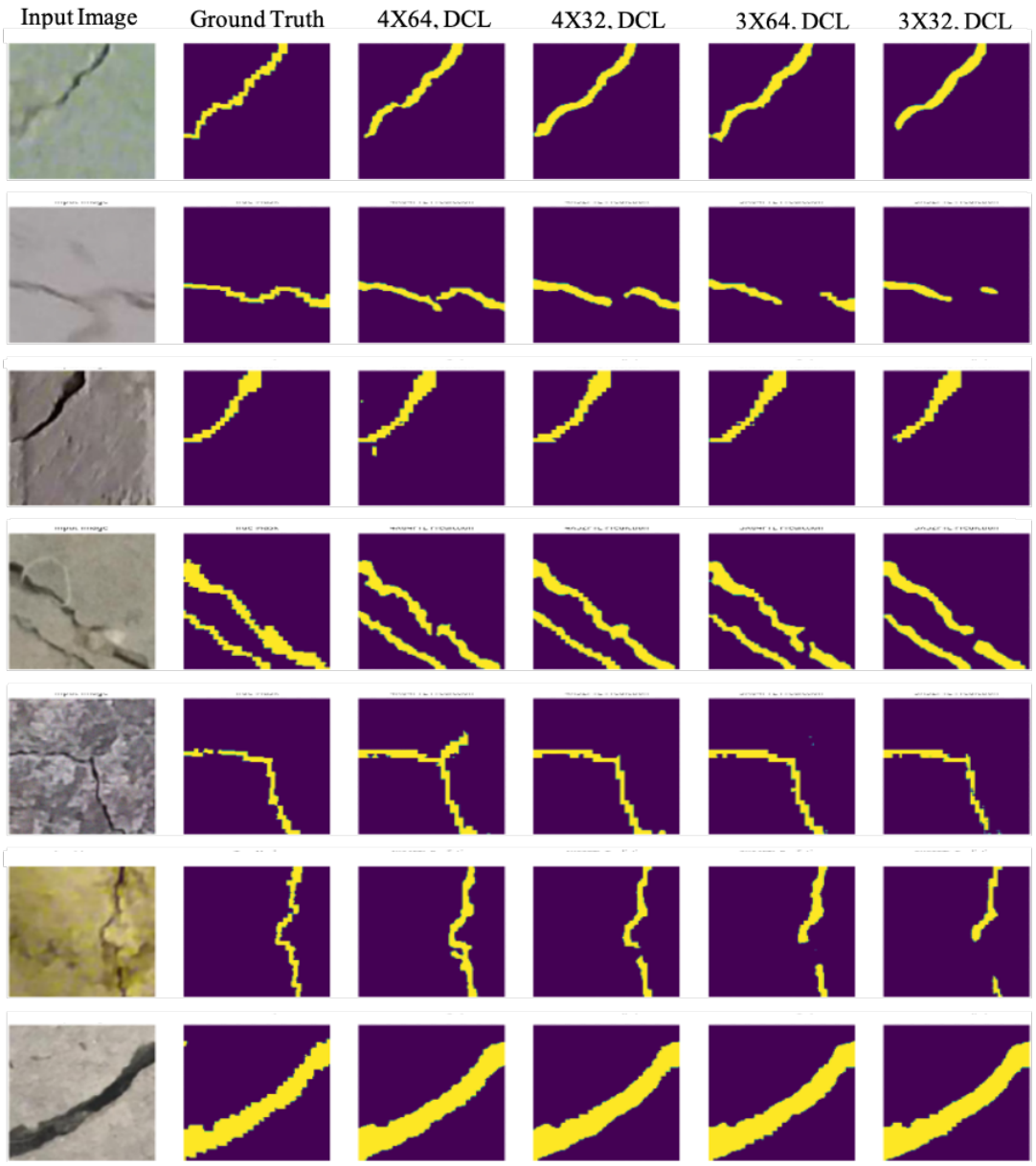


Figure 4.11: *Comparison of different model architectures trained with DCL*

Table 4.4 compares the computational cost of different architectures in terms of mean processing time on a 128 × 128 sub image. The best result is marked with bold font.

Table 4.4: *Computational costs of models*

Model	Number of Parameter	Processing time per sub-image (s)
Unet-4x64	310,328,37	0.5197
Unet-4x32	7,760,645	0.1808
Unet-3x64	7,698,437	0.4018
Unet-3x32	1,926,149	**0.1475**

4.6 Summary

Theoretically, a larger CNN architecture usually has more potential to finish a certain task than a smaller one. But it's unnecessarily true that a larger model always reaches a more satisfying result. It can be concluded from Table 4.2 that with the training setup and data-set specified in Section 4.2 and Section 4.5, the best model can reach 88.36% for sensitivity. The performance of models is also influenced by different crack image types. Thin cracks, blurred images, cracks with complicated shapes and the presence of noise can all have negative influence to the final prediction at different extents. On the other hand, a larger architecture costs more training time and computational power in the application. Comparing Unet-4x64 and Unet-4x32 illustrated in Table 4.4, more than 65% of the processing time is saved by decreasing the number of feature maps.

From both training curves (Figure 4.8) and visualized training progression (Figure 4.9) it can be concluded that DCL, TL and FTL can make the models focus more on the positive predictions and thus increase the sensitivity to crack pixels. While the recognition ratio of true positive pixels is improved, false positive results are also increased and cause decrease for the specificity. Comparing Table 4.2 and Table 4.3, it is obvious the benefit gained from these generalized loss functions is more considerable than the compromise in specificity, especially for the thin crack cases.

Overall, the Unet-4x32 trained with DCL is proposed to be used for the sub-image crack segmentation task, which not only has a stable performance for different test groups and the highest sensitivity on the test data-set, but also save considerable computational power.

5

Experiment

As introduced in Section 1.5, the proposed algorithm is a combination of the classification and segmentation CNNs. In this chapter, the performance of this combined algorithm is tested on photo of cracks.

The algorithm is first tested on the photos collected from real structures by a hand-held camera, which has a resolution of 1836x3264 pixels. These photos are also the data source of the sub-images used for training of the two CNN models. The image classification and segmentation result are shown in Figure 5.1, where the sub-images classified as positive are marked with red squares and the pixel-level segmentation highlights the cracks with red color. Since the accuracy of CNN models are always increased on training data, the test performed on these photos is just to show how the combined algorithm works.

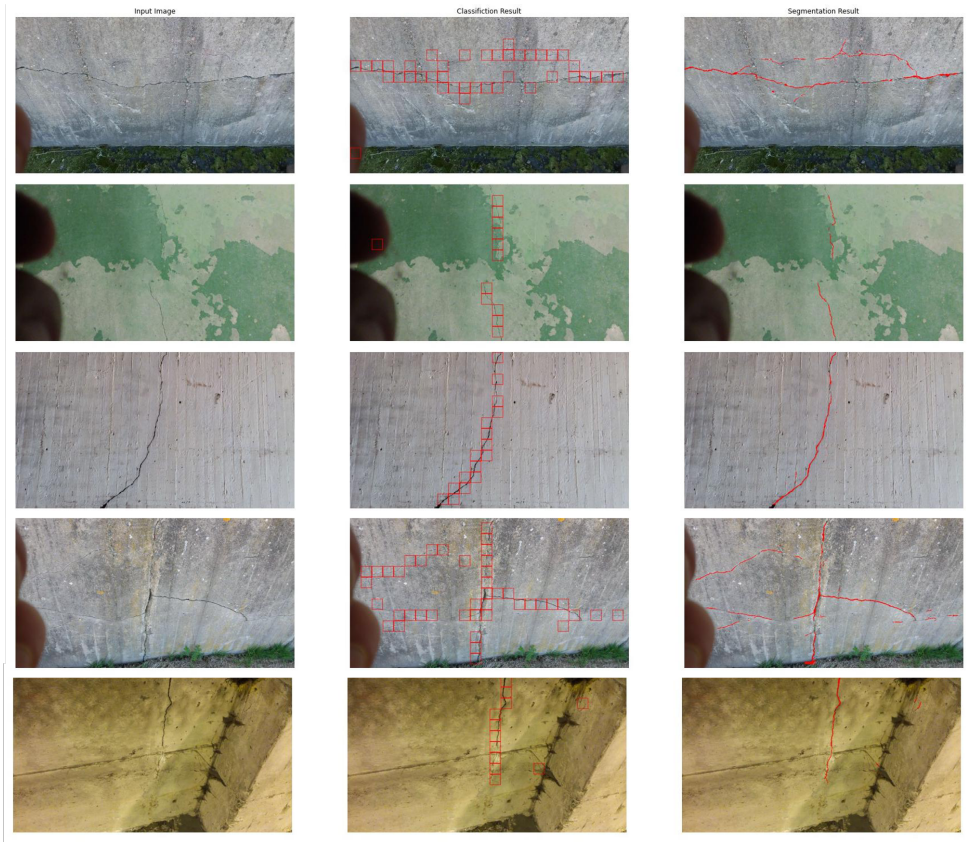


Figure 5.1: *Full image test*

To learn the performance of the proposed algorithm and the feasibility of this method in the field application, a experimental inspection is conducted with a drone. In the field situation, to get a clear photo of thin cracks, the drone needs to get very close to the structure and the camera lens need to have a long focal length. DJI Mavic2 Zoom (Figure 5.2) is chosen to perform this inspection. The powerful omnidirectional obstacle sensing system and the 2x optical zoom function make it possible to shoot cracks from a safety distance of approximately 1.5 3m. The specifications of this drone model and the camera are listed in Table 5.1

Figure 5.2: *DJI Mavic2 Zoom*

Table 5.1: *Specification of the Drone and Camera*

Drone Model	DJI Mavic2 Zoom
Takeoff Weight	905 g
Max Speed	20 m/s
Max Wind Resistance	38 kph
Lens	24-48 mm (35 mm Format Equivalent)
Camera sensor	1/2.3" CMOS
Image Size	4000Œ3000
Image Format	.JPG

The inspected objects are a wall in the structure lab and a damaged concrete beam specimen (Figure 5.3). The wall is inside a building and the drone has to be operated at a indoor environment. A minimum distance of 1.5m from the wall can be reached. The left photo in Figure 5.3 also shows the operation of the drone.

Figure 5.3: *The damaged wall and concrete specimens*

The wall experienced a crush by a truck and multiply cracks can been seen on the surface. It is a brick wall but the rough surface has similar features with a concrete wall. Figures 5.4 shows the photos taken by the drone and the crack detection results of the algorithm. The cracks has been very well detected but there are also false positive detection from both classification and segmentation procedures. These false identifications mostly occurs at the edges of the bricks. This is most likely cased by the fact that all the training data for both CNNs is collected from concrete structures and the algorithm is relatively vulnerable to the brick wall situation.

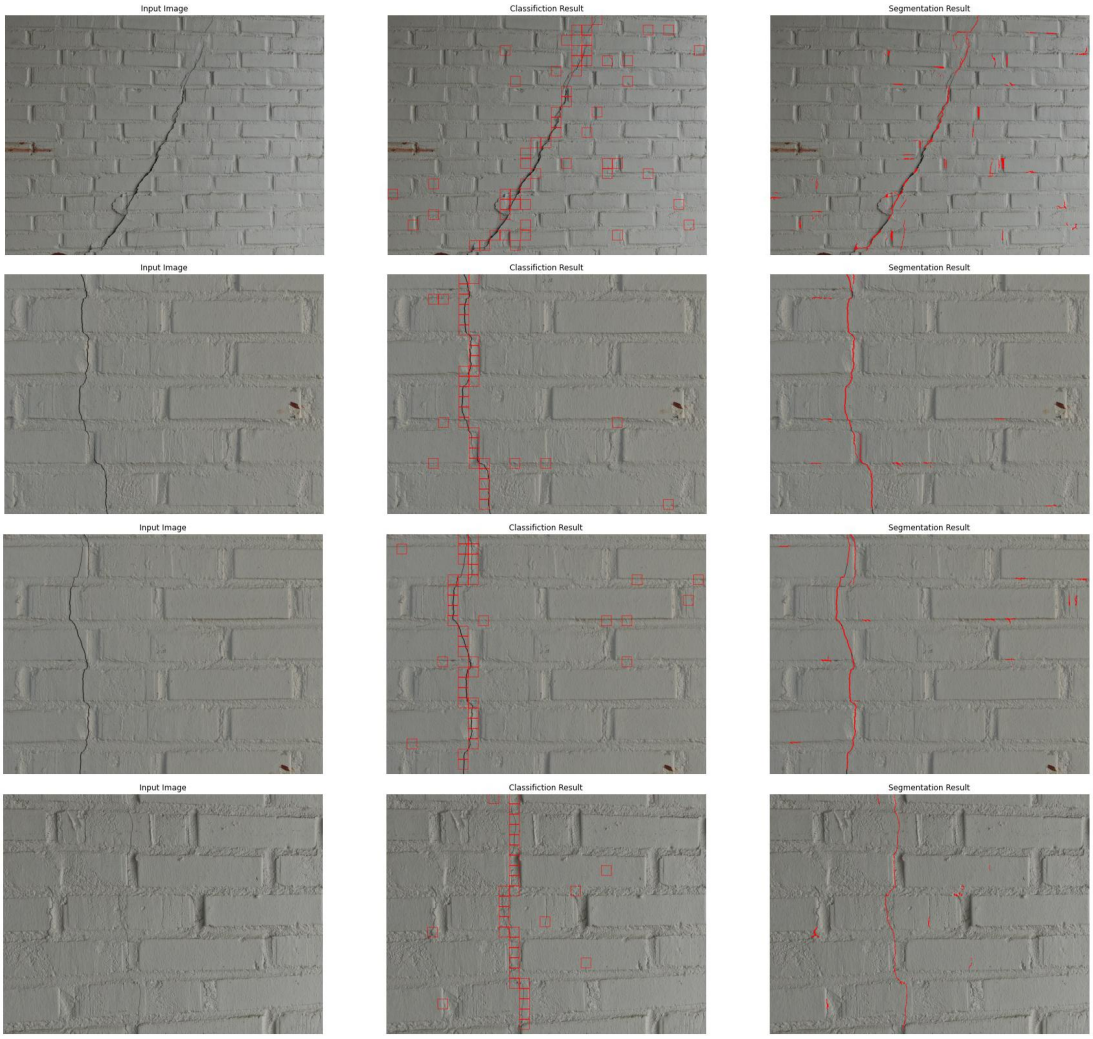


Figure 5.4: *Images of the damaged wall and crack detection results*

The specimen is taken from Gutenberg harbor for research purpose. The specimen is severely damaged and very wide cracks can be found at the mid-span. Some marks on the surface indicating the location of the cracks have been made. As can be seen from Figure 5.5, these marks cause some false positive prediction, while on the same photo, the algorithm still managed to recognize the cracks. A very wide crack appears in the third photo and the algorithm fail to detect it. The reason of this failure is likely to be that the width of the crack in the image exceeds the sub-image size.

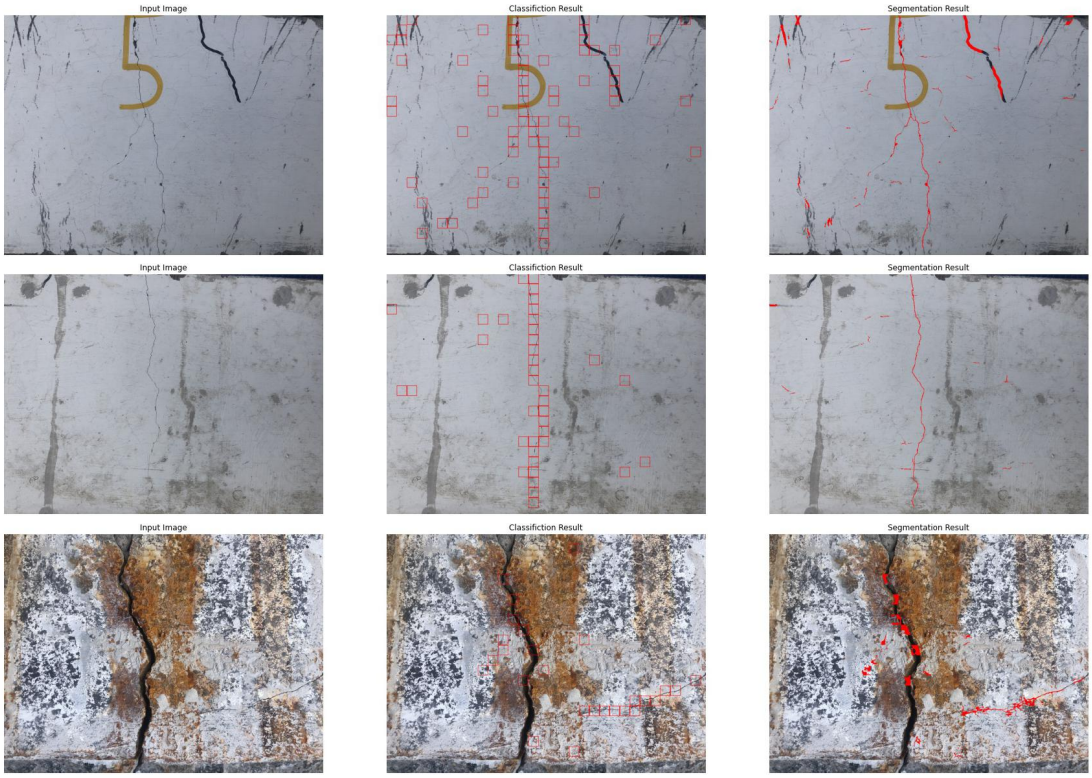


Figure 5.5: *Images of the damaged specimen and crack detection results*

6

Conclusion and prospect

6.1 Conclusion

This project proposes an image-based crack detection algorithm that can achieve automatic crack extraction without any human intervention. Two CNN models are used for classification and segmentation respectively. The whole prediction process which includes both classification and segmentation will only take 86 seconds in total. The visual performance on the whole scale testing image indicates the high accuracy of both CNN intuitively. This project has shown the practicability of applying the deep learning method into the structural health monitoring field.

The biggest challenge in this project is the variety of the image database. This database is very raw and realistic of how a drone token close-up photo will look like. The large image variation will debase the final prediction accuracy to some degree without doubly. In the tough process of searching for the "best CNN model", a lot of experiments have been tried out. However, the conclusion is that transfer learning can actually give a quick first glance of how suitable the designed deep CNN network can work on the cross-domain data-set. A more efficient way of finding suitable network architecture can reach by applying the transfer learning method firstly, then the appropriate parameter customized for each specified working case should be sought by carrying out different experiments. For example, in this project, the alpha value representing the number of the channel has been modified and various loss function has been tried out. The series of loss functions rooted in weighed cross-entropy has shown superiority when the data-set is highly imbalanced. Finally, small ad-justification like the trade-off between computational power and prediction performance indicator can be made based on the priority for different environments.

Despite the good results on both the numerically side and visual effect side. The training effort still should be counted. In this project, both CNN have a long process for training if one pursues real-time crack detection. Moreover, the network trained on the crack image not necessarily has high accuracy on the testing image if the testing image differs a lot from the original training image, which might suggest the network should keep training when it goes for industrial usage. The limitation of this automated crack detection algorithm is that the classification step is decisive to the overall performance, which will cause vulnerability once the classifier pronounces false positive or false negative. One of the optimizing directions is to

reducing the probability threshold for positive admission to decrease the false negatives then developing a dapper algorithm based on the crack geometry feature after the segmentation step to filter out the increased false positives.

6.2 Prospect

As mentioned in the introduction, the proposed algorithm serves as a part of the Digital Twin concept. To achieve the target of SHM based on digital representations of real structures, the proposed crack detection algorithm should be integrated properly into the work flow of Digital Twin.

Data acquisition:
The proposed algorithm takes digital images as input. Drones with high resolution camera provides a highly efficient way to perform visual inspection and image data acquisition. Different country has different regulations about the distance when one flies a drone above the infrastructure. But it all needs the operator to find cracked religion from a distance. With respect to this demand, the object detection technique might be the solution. There has been a lot of researches built the object detection method[35] in drone and located the cracked area successfully. By flying the drone to the target location, the drone can bring the close-up crack pictures back.

Figure 6.1: *One example for applying object detection method to locate the cracked area in concrete bridge*

The finite element model:
In order to map the cracked sub-image into the bridge model, the 3D cloud technique might be used. This means the image should contain its own location information or relative location information in the first place. To successfully utilize the crack detection results, this mapping process needs to be accurate as well.

Bibliography

[1] Yuequan Bao, Zhicheng Chen, Shiyin Wei, Yang Xu, Zhiyi Tang, and Hui Li. The State of the Art of Data Science and Engineering in Structural Health Monitoring. *Engineering*, 5(2):234–242, 2019.

[2] Elisa Negri, Luca Fumagalli, and Marco Macchi. A review of the roles of digital twin in cps-based production systems. *Procedia Manufacturing*, 11:939–948, 2017. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

[3] Yi-Chen Zhu, David Wagg, Elizabeth Cross, and Robert Barthorpe. Real-Time Digital Twin Updating Strategy Based on Structural Health Monitoring Systems. In Zhu Mao, editor, *Model Validation and Uncertainty Quantification, Volume 3*, pages 55–64, Cham, 2020. Springer International Publishing.

[4] Abdulmotaleb El Saddik. Digital Twins: The Convergence of Multimedia Technologies. *IEEE MultiMedia*, 25:87–92, 2018.

[5] Sandeep Sony, Shea Laventure, and Ayan Sadhu. A literature review of next-generation smart sensing technology in structural health monitoring. *Structural Control and Health Monitoring*, 26(3):e2321, 2019.

[6] Sandeep Sony, Kyle Dunphy, Ayan Sadhu, and Miriam Capretz. A systematic review of convolutional neural network-based structural condition assessment techniques. *Engineering Structures*, 226(August 2020):111347, 2021.

[7] Lei Zhang, Fan Yang, Yimin Daniel Zhang, and Ying Julie Zhu. Road crack detection using deep convolutional neural network. In *2016 IEEE international conference on image processing (ICIP)*, pages 3708–3712. IEEE, 2016.

[8] Byunghyun Kim and Soojin Cho. Automated Vision-Based Detection of Cracks on Concrete Surfaces Using a Deep Learning Technique. *Sensors*, 18(10), 2018.

[9] Fangzheng Lin, Jiesheng Yang, Jiangpeng Shu, and Raimar J Scherer. Crack semantic segmentation using the u-net with full attention strategy. *arXiv preprint arXiv:2104.14586*, 2021.

[10] Lingxin Zhang, Junkai Shen, and Baijie Zhu. A research on an improved unet-based concrete crack detection algorithm. *Structural Health Monitoring*, page 1475921720940068, 2020.

[11] Ikhlas Abdel-Qader, Osama Abudayyeh, and Michael E Kelly. Analysis of edge-detection techniques for crack identification in bridges. *Journal of Computing*

in Civil Engineering, 17(4):255–263, 2003.

[12] Tomoyuki Yamaguchi, Shingo Nakamura, Ryo Saegusa, and Shuji Hashimoto. Image-based crack detection for real concrete surfaces. *IEEJ Transactions on Electrical and Electronic Engineering*, 3(1):128–135, 2008.

[13] Takafumi Nishikawa, Junji Yoshida, Toshiyuki Sugiyama, and Yozo Fujino. Concrete crack detection by multiple sequential image filtering. *Computer-Aided Civil and Infrastructure Engineering*, 27(1):29–47, 2012.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[15] Arun Mohan and Sumathi Poobal. Crack detection using image processing: A critical review and analysis. *Alexandria Engineering Journal*, 57(2):787–798, 2018.

[16] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[17] Kunihiko Fukushima and Sei Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern recognition*, 15(6):455–469, 1982.

[18] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[19] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, may 2017.

[21] Andrej Karpathy, FF Li, and J Johnson. Cs231n convolutional neural networks for visual recognition (2016). *URL http://cs231n. github. io*, 50, 2017.

[22] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[23] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *arXiv preprint arXiv:1805.11604*, 2018.

[24] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[25] Young-Jin Cha, Wooram Choi, and Oral Büyüköztürk. Deep learning-based

crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, 32(5):361–378, 2017.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[28] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[30] Cao Vu Dung et al. Autonomous concrete crack detection using deep fully convolutional neural network. *Automation in Construction*, 99:52–58, 2019.

[31] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[32] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571, 2016.

[33] Seyed Raein Hashemi, Seyed Sadegh Mohseni Salehi, Deniz Erdogmus, Sanjay P Prabhu, Simon K Warfield, and Ali Gholipour. Tversky as a loss function for highly unbalanced image segmentation using 3d fully convolutional deep networks. *arXiv preprint arXiv:1803.11078*, 2018.

[34] Nabila Abraham and Naimul Mefraz Khan. A novel focal tversky loss function with improved attention u-net for lesion segmentation. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 683–687, 2019.

[35] Chaobo Zhang, Chih-chen Chang, and Maziar Jamshidi. Concrete bridge surface damage detection using a single-stage detector. *Computer-Aided Civil and Infrastructure Engineering*, 35(4):389–409, 2020.

DEPARTMENT OF ARCHITECTURE AND CIVIL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY