EXPLOITATION OF DEEP LEARNING IN THE AUTOMATIC DETECTION OF CRACKS ON PAVED ROADS

WON MO JUNG

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN EARTH AND SPACE SCIENCE

YORK UNIVERSITY

TORONTO, ONTARIO

JULY 2020

Abstract

Information on the severity of pavement cracks is critical for pavement repair services. This study aimed to exploit the applications of deep learning networks to improve the detection and classification of pavement cracks. An improved Convolution Neural Network (CNN) with structured prediction was proposed. Also, Fully Convolutional Network (FCN), U-Net, and attention U-Net were implemented and explored with different optimizers and loss functions. The developed networks were tested on the data collected in Ontario, Canada with the purpose of localizing the cracks and identifying their severity levels based on three categories (low, medium, and high). The results showed that the improved CNN approach performed better than its original version with the F1-score increased from (5.21%, 30.85%, 83.51%) to (19.63%, 55.60%, 85.89%) for the detection of the cracks with the three severities. Furthermore, FCN, U-Net, and attention U-Net achieved slightly better results than the improved CNN approach with the F1-scores of (32.08%, 68.82%, 89.89%), (40.06%, 69.97%, 89.07%), and (40.53%, 71.27%, 89.95%), respectively.

Acknowledgements

I would like to express my deep gratitude to my supervisor, Dr. Baoxin Hu, for her patient guidance and meaningful critiques of this research. Without her support and guidance, it would have been difficult for me to complete this research in the field of deep learning.

I would like to express great appreciation to Dr. Jian-Guo Wang, Dr. Gunho Sohn, and Dr. Ruth Urner for their valuable suggestions on this research. Their expertise in Image Processing, Computer Vision and Machine learning guided me with different perspectives.

Furthermore, I would like to thank Natural Sciences and Engineering Research Council (NSERC) and Ministry of Transportation Ontario (MTO) for their financial support. Special thanks to Gideon Gumisiriza at MTO for providing the necessary data for training and testing purposes.

Finally, I would like to extend my gratitude to my friends and lab colleagues for their enthusiastic encouragements and moral supports.

Achievements

1. AOLS (Association of Ontario Land Sueyveyors) 2019 Graduate Poster Competition Award. The poster titled “Automatic Road Crack Detection and Localization with Identification of Pavement Distress Levels using Laser Ranges“ by “Won Mo Jung and Faizaan Naveed“ was selected as 4th best graduate student poster presentation.

2. 40th Canadian Symposium on Remote Sensing & Geomatics Atlantics (CSRS) 2019. The presentation titled “Automatic Road Crack Detection and Localization with Identification of Pavement Distress Levels using Laser Ranges“ by “Won Mo Jung was selected as 2019 3rd place best oral presentation award.

3. A published work in the journal Geomatica: Jung, W., F. Naveed, B. Hu, J. Wang, and N. Li, 2019, “Exploitation of deep learning in the automatic detection of cracks on paved roads”, Geomatica, dx.doi.org/10.1139/geomat-2019-0008.

Abbreviations

2-D: 2-Dimensions

ADABOUND: Adaptive Gradient Methods with Dynamic Bound

ADAM: Adaptive Moment Estimation

AG: Attention Gates

ARAN: Automatic Road Analyzer

CNN: Convolution Neural Network

DSC: Dice Coefficient

FCN: Fully Convolution Network

FN: False Negative

FP: False Positive

FTL: Focal-Tversky Loss

LCMS: Laser Crack Measurement System

MTO: Ministry of Transportation Ontario

ReLU: Rectified Linear Unit

SGD: Stochastic Gradient Descent

Tanh: Hyperbolic Tangent

TN: True Negative

TP: True Positive

VGG: Visual Geometry Group from Oxford

Table of Contents

List of Tables

List of Figures

Chapter One: Introduction

Due to traffic, environmental factors, and aging, road pavements usually experience different types of distress and deterioration that include cracks, surface defects, and profile deformation (McGhee, 2004; Miller et al., 2003; Bennett et al., 2007). The qualities of pavements are usually characterized by distress type, severity, and extent. Effective and accurate monitoring of the pavement condition is paramount to determine if it provides a comfortable, safe, and efficient service to the public. It also assists the road management authority to make decisions on appropriate maintenance and rehabilitation. Traditionally, pavement surveys involve observing and recording surface defects and degradation through walking or slowly driving over pavements (asphalt or concrete) by certified professionals. The manual inspection is usually time-consuming, and the collected pavement information is subject to the individual surveyor's level of experience and knowledge (Jing et al., 2010). Manual techniques are generally considered labor-intensive, slow, expensive, and sometimes unsafe due to the traffic. The early efforts to develop automatic or semi-automatic systems for pavement assessment are mainly image-based. The qualities of the images and software for data processing and analysis limit their operational employment (Cafiso et al., 2006;

Teomete et al., 2005; Yu et al., 2007; Jing et al., 2010).

With the development of laser line projectors and custom optics (Cafiso et al., 2017), the quality of collected data has been improved, but the costs of the systems and their operations limit their usage. As an example, the Ministry of Transportation of Ontario (MTO) currently owns such systems to inspect the pavement conditions, but due to their high cost, these systems are mainly operated on major highways and roads with lots of traffic. Hence, the recent advance in imaging technologies and artificial intelligence provides a good opportunity to improve the performance of the image-based system for the assessment of pavement conditions.

Several attempts have been made in the past decade for automatic crack detection based on machine learning techniques (Ouyang et al., 2010; Davis, 2011; Oliveira et al., 2013; Kapela et al., 2015; Shi et al., 2016). In general, these existing techniques require extensive pre-processing due to irregularities on the crack surface, variations in crack texture, and non-uniform illuminations. Although reasonable results are obtained through machine learning techniques (Kapela et al., 2015), they are very dependent on the radiometric and geometric qualities of the original images and the corresponding pre-processing techniques. Moreover, traditional machine learning techniques do not always achieve high accuracies in the localization

of cracks (approximately 60% accuracy) and tend to over-estimate the widths of cracks (Oliveira et al., 2013). In addition, many methods are not able to distinguish cracks with different levels of severity, which prevent the detection results from being used in an automated system for road repair services (Davis, 2011).

Recent advance in deep learning provides a good opportunity to improve the detection of pavement cracks. Fundamentally, deep learning is a subset of machine learning that functions similarly but has different capabilities. In general, traditional machine learning models require some human interactions to return an accurate prediction, whereas deep learning models are designed to logically comprehend the data on its own as to how a human would understand the data. There are various types of deep learning models that are designed for specific tasks. Hence, designing a new deep learning architecture requires an exetensive knowledge about both deep learning and traditional machine learning to fully construct an appropriate network. Similarly, even the users need prior knowledge to choose the appropriate deep learning models to properly accomplish the desired tasks. The most trending type of neural network in image recognition tasks is the convolution-based neural networks. Convolution-based neural networks take advantage of local spatial coherence of images to reduce the number of

operations needed to process an image by using convolution operation on patches of adjacent pixels. Based on the aspect of convolution operations, different types of convolution-based networks such as Convolution Neural Network (CNN), Fully Convolution Network (FCN) (Long et al., 2015), U-Net (Ronneberger et al., 2015), and Attention U-Net (Oktay et al., 2018) are introduced in recent years. Although all these networks use convolution layers to extract features, their capabilities vary depending on the structure of the network. For instance, a basic CNN is designed to simply classify an image into different categories, but other networks, such as FCN and U-Net can not only classify, but also localize the objects within the image. Therefore, FCN and U-Net are often referred to as segmentation neural networks. However, these networks that can perform segmentation of an image are often computationally intensive to be trained.

In recent years, more and more open-source packages are becoming available for designing and building a deep neural network for various classification tasks. Nowadays, graphic cards that are currently available to the public have enough computing power to construct and train a neural network. In the context of the detection of pavement cracks, Yusof et al. (2018) applied a CNN to detect cracks; the whole image was divided into a series of grids and the detection was done on individual grids. However, the resulting binary

image was very coarse and inadequate for pixel-level evaluation. Likewise, Fan et al. (2018) developed CNN with a structured prediction for crack detection. To my best knowledge, this network is the state-of-art of deep learning networks in pavement crack localization. Instead of using the whole image as the input, Fan et al. (2018) employed a small image patch of size 27x27 as the input to predict a 5x5 binary map of the corresponding center of the input image patch. By repeating this procedure with a sliding window, a binary map of the whole pavement image was acquired. Even though CNN with a structured prediction proposed by Fan et al. (2018) performed extremely well on localizing the cracks on pavement images, it did not distinguish cracks with different severities.

The goal of this research was to fully exploit various convolution-based deep learning networks to improve the detection of cracks on paved roads and classify different levels of severity on the localized cracks. To achieve this goal, the following two objectives were fulfilled in this thesis research:

(1) Improve the CNN with a structured prediction

CNN with a structured prediction proposed by Fan et al (2018) was improved

in this study in the following two aspects. As mentioned earlier, the network by Fan et al. (2018) was only able to distinguish crack and non-crack pixels, hence it was improved in this study to classify cracks with three levels of severities. In addition, even though the structured prediction proposed by Fan et al. (2018) accounted for the spatial correlation between the center pixel and its surrounding pixels in an image patch, these pixels were treated with equal weights in the loss function. In this study, a weighted loss function was proposed to effectively distribute the weights with respect to the spatial location of the pixels.

(2) Exploit FCN, U-Net, and attention U-Net in crack detection

Although the improved CNN with a structured prediction can produce a segmentation map of a pavement image, there were few limitations. Since the output from this network was a small image patch, these patches had to be stitched together to form a whole pavement image, which slowed down the prediction process by a huge factor. To overcome this, convolution-based segmentation approaches such as FCN (Long et al., 2015), U-Net (Ronneberger et al., 2015), and Attention U-Net (Oktay et al., 2018) were implemented in this study to detect pavement cracks for the first time.

This thesis consists of six chapters. After the current introduction, the second chapter describes the background of deep learning approaches including the fundamentals of deep learning networks, such as optimizers, loss functions, and the commonly used networks, such as CNN, FCN, U-Net, and attention U-Net. Methodologies developed in this research and the results obtained are provided in Chapters three and four, respectively. Chapter five discusses the developed methods in detail. Conclusion and future work are summarized in Chapter six.

Chapter Two: Background

2.1 Fundamentals to deep learning networks

Within the past decade, deep learning techniques have become one of the most trending methods in classification tasks, due to a large amount of data and publicly available open-source code packages. Unlike traditional machine learning approaches, deep learning techniques are data-driven. In a typical machine learning algorithm, the explicit extraction and selection of features are needed for a given task. However, in the deep learning algorithms, features are automatically extracted and selected based on the provided data and the corresponding labels during the training stage. In addition, one of the biggest reasons that deep networks have recently become so popular is that the current GPUs now have enough computational capability to implement effective feature extraction such as 2-dimension (2D) convolutions within the neural network. Consequently, large datasets can be handled, which directly leads to better-trained networks.

Most of the networks including the commonly used convolution-based networks are known as the feed-forward neural network, because the output of a layer becomes the input of the following layer, and the output from a layer is never fed back into the previous layer. The feed-forward neural networks are

generally composed of an input layer, hidden layers, and an output layer. The input layer and output layer are where the data is input to the network, and the result is output from the network. The hidden layer is a new concept introduced in neural networks. In neural networks, the layers between the input layer and the output layer are referred to as hidden layers. These hidden layers are responsible for manipulating the data to get the desired output. These layers are called hidden layers because they are implicitly available and are private to the network. Likewise, CNN is also composed of the input layer, hidden layers, and the output layer. In CNN, the convolution layers are considered as the hidden layers between the input layer and the output layer.

Furthermore, the neural networks are generally trained through a process called backpropagation. Backpropagation is an algorithm that traces back from the output of the model, through the different neurons which were involved in generating that output, back to the original weight applied to each neuron. Backpropagation adjusts the weight for each neuron to minimize the difference between the true label and the predicted label.

Each hidden layer is equipped with an activation function. Activation functions are mathematical equations that determine the output of the layer. It determines whether the corresponding input should be activated or not. In deep learning networks, sigmoid, hyperbolic tangent, Rectified Linear Unit

(ReLU), and softmax functions are commonly used as an activation function. One of the most popular ways to model a neuron's output as a function of its input x is with the sigmoid function defined in (2.1.1).

$$f(x) = (1 + e^{-x})^{-1} \qquad (2.1.1)$$

The output of the sigmoid function ranges from 0 to 1, which is shown in Figure 2.1.1.

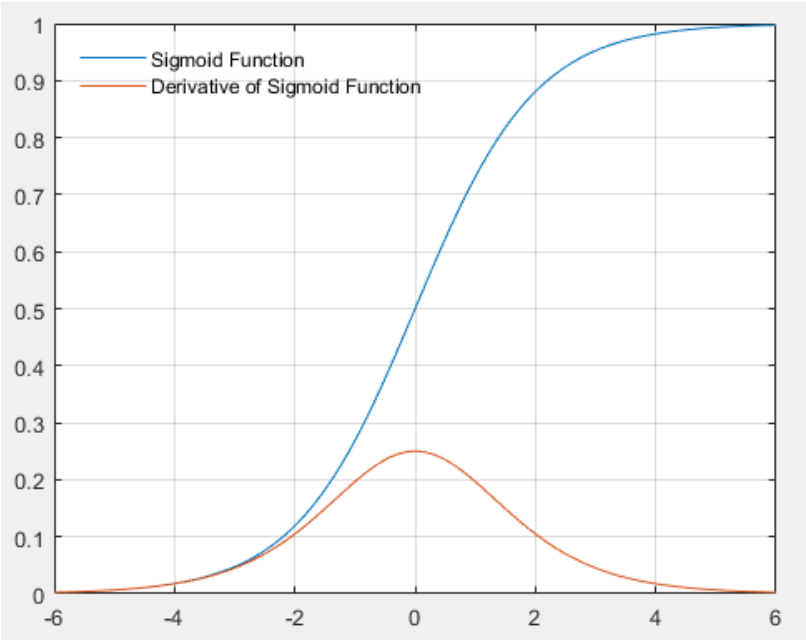


Figure 2.1.1 Sigmoid function and derivative of the sigmoid function

Due to its simplicity, this function has become one of the widely used

activation functions. However, the sigmoid function has a disadvantage of having a vanishing gradient problem. A gradient is produced during the backpropagation phase of the network and it accumulates throughout the network by multiplying the derivatives of each layer. Therefore, the vanishing gradient problem occurs when the derivative of the layer's output with respect to its input is too small. As shown in Figure 2.1.1, the derivative of the sigmoid function with respect to its input results in a small output value when the input value is big. Thus, the gradient ends up being a small value and when these small gradients are stacked, the resulting gradient gets exponentially smaller. Therefore, the network does not update its weights properly and leads to a non-convergence. Since the sigmoid function has an output value ranging within [0,1], it can potentially lead the gradient updates to vary in different directions. Thus, this range of output makes the optimization harder as it does not compensate for the other directions. Additionally, the sigmoid function has a slower convergence than other activation functions.

To accommodate the optimization issue on sigmoid, the hyperbolic tangent function shown in (2.1.2) is often used as a replacement.

$$f(x) = \tanh(x) \qquad (2.1.2)$$

Unlike the sigmoid function, the hyperbolic tangent function's output is centered at zero and its value ranges from -1 to 1. The hyperbolic function is

shown in Figure 2.1.2. Since the output ranges from -1 to 1, the optimization is more stable and much easier than a sigmoid function. However, the hyperbolic tangent function also suffers from the vanishing gradient problem.

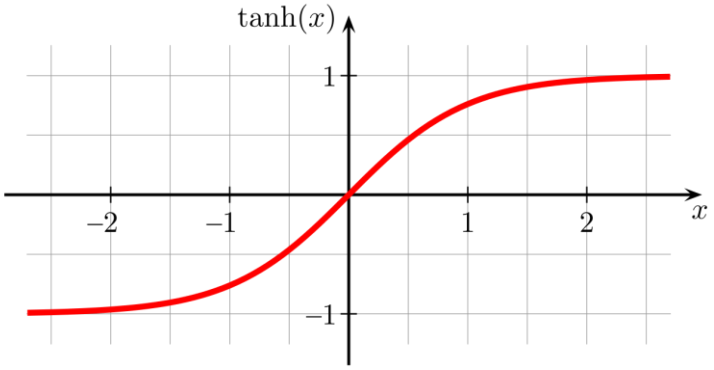


Figure 2.1.2 Hyperbolic Tangent function

Therefore, many researchers use ReLU (Nair et al., 2010) to avoid the vanishing gradient problem. Just like any other activation functions, ReLU is also differentiable and introduces non-linearity to the feature maps. ReLU is defined in (2.1.3).

$$f(x) = \max(0, x) \qquad (2.1.3)$$

ReLU simply defines if input $x < 0$, then $f(x) = 0$ and if $x \geq 0$, then $f(x) = x$. The ReLU function is shown in Figure 2.1.3. Aside from solving the vanishing gradient problem, ReLU trains the network several times faster than

the hyperbolic tangent function (Krizhevsky et al., 2012). Consequently, researchers generally use ReLU as an activation function after each set of convolution layers. Since the output from ReLU is not scaled, ReLU can only be used within hidden layers.

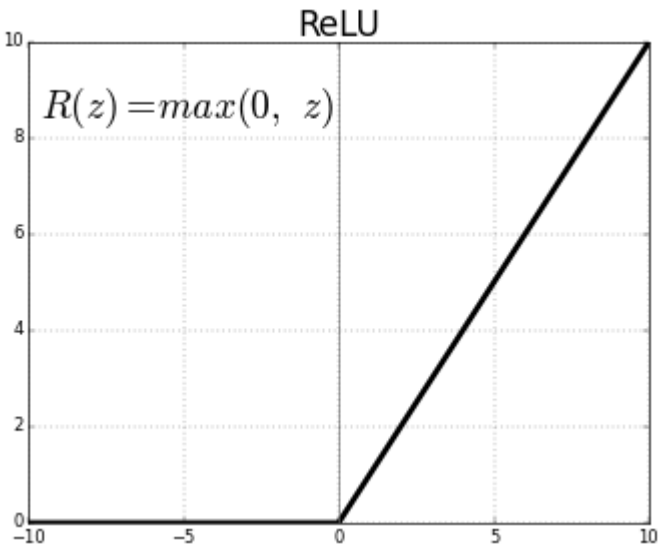


Figure 2.1.3 Rectified Linear Unit (ReLU)

Unlike the other activation functions, softmax function is applied at the last hidden layer for classification tasks. In multi-class classification tasks, the softmax function defined in (2.1.4) is used.

$$\sigma(x)_i = \frac{exp(x_i)}{\sum_{j=1}^{K} exp(x_j)} \qquad (2.1.4)$$

A softmax function is a form of logistic function that normalizes an input value into a vector of values that follows a probability distribution that sums up to 1. Similar to the sigmoid function, the output values are between the range of 0 and 1. Therefore, the softmax function is suitable for multi-class classification tasks. However, the sigmoid function is rather preferred in multi-label classification tasks, because the softmax function is not designed to output multiple positive predictions.

During the training phase, neural networks are trained to find an appropriate weight for each activated neuron through weight updates. The neural network uses a loss function to measure the discrepancy between the model's prediction and the ground truth. Then, the network adjusts the weight for each neuron to predict the correct label. The network urges to minimize the loss function and through backpropagation, the network adjusts the weight assigned to each neuron to generate a prediction that is close to the ground truth. Depending on the dataset and the application, the choice of loss function varies. In this study, cross-entropy, dice, and Tversky loss function is exploited.

In a multi-class classification problem, a cross-entropy loss function is generally used to penalize the deviation of predicted probability p_c of class c

from the ground truth y of class c. The cross-entropy loss function is defined by (2.1.5):

$$L = -\sum_{c=1}^{M} y_c \log(p_c) \qquad (2.1.5)$$

Cross-entropy loss function performs well on most of the classification tasks, but it does not have any variables to control the class imbalance dataset.

If the dataset suffers from a class imbalance problem like in medical image analysis, where the region of interest is usually found in a very small fraction of the full image, the network often has trouble generalizing the data. A common method to reduce the effects of class imbalance is to use the dice coefficient (DSC) defined in (2.1.6). The DSC is a measure of overlap that is widely used to evaluate segmentation performance (Sudre et al., 2017).

$$DSC_c = \frac{\sum_{i=1}^{N} p_{ic} g_{ic} + \epsilon}{\sum_{i=1}^{N} p_{ic} + g_{ic} + \epsilon} \qquad (2.1.6)$$

$$DL_c = \sum_{c} 1 - DSC_c \qquad (2.1.7)$$

(2.1.6) indicates the dice score coefficient where p_{ic} and g_{ic} represents predicted label and ground truth label for each class c. The ϵ simply provides numerical stability to prevent division by zero. By subtracting the dice score

coefficient from 1 for each class, dice loss is acquired as shown in (2.1.7).

Although the dice loss function is commonly used for an imbalanced dataset, it still does not differentiate the weights given between false positive and false negative detections. Consequently, the segmentation maps result in high precision with low recall. When the region of interest is extremely small with highly imbalanced data, assigning more weights on the false negative detections than false positive detections improve the recall rate. Therefore, the Tversky similarity index (Abraham et al., 2019) is introduced to generalize the dice score, which allows flexibility between false positive and false negative, as shown in (2.1.8).

$$TI_c = \frac{\sum_{i=1}^{N} p_{ic} g_{ic} + \epsilon}{\sum_{i=1}^{N} p_{ic} g_{ic} + \alpha \sum_{i=1}^{N} p_{i\bar{c}} g_{ic} + \beta \sum_{i=1}^{N} p_{ic} g_{i\bar{c}} + \epsilon} \qquad (2.1.8)$$

From (2.1.8), p_{ic} indicates the probability that pixel i being true positive and $p_{i\bar{c}}$ is the probability pixel i being the false positive. The same is true for g_{ic} and $g_{i\bar{c}}$, respectively. Hyperparameters α and β are used to shift the emphasis to improve recall in the case of large class imbalance. Similar to dice loss, the Tversky loss function is used by minimizing (2.1.9).

$$TL_c = \sum_{c} 1 - TI_c \qquad (2.1.9)$$

As the small region of interest does not contribute to the loss function

significantly, the dice loss function is not always suitable for segmentation tasks. To avoid this issue, researchers use a focal loss function (Lin et al., 2017), which reshapes the cross-entropy loss function with a modulating exponent to down-weight errors assigned to well-classified examples. The main advantage of using focal loss is to prevent the vast number of easily defined negative examples from dominating the gradient to alleviate class imbalance effect. Hence, the focal Tversky loss function (Abraham et al., 2019) is used in segmentation tasks, where it is parameterized by γ to control between easy background and hard region of interest from training examples. The Focal Tversky Loss (FTL) is defined in (2.1.10).

$$FTL_c = \sum_c (1 - TI_c)^{\frac{1}{\gamma}} \qquad (2.1.10)$$

If a pixel is misclassified with a high Tversky index, the FTL is unaffected, but if a pixel is misclassified with a small Tversky index, FTL is decreased significantly. When $\gamma > 1$, the loss function concentrates more on the less accurate prediction that has been misclassified. The increasing values of the Tversky index are mapped to flatter regions of the FTL curve with increasing values of γ and shown in Figure 2.1.4. As shown from Figure 2.1.4, the over-suppression of the FTL is observed when the class accuracy is high and as the model becomes close to convergence. Thus, Abraham et al. (2019) concludes

that the best performance is achieved when $\gamma = \frac{4}{3}$.

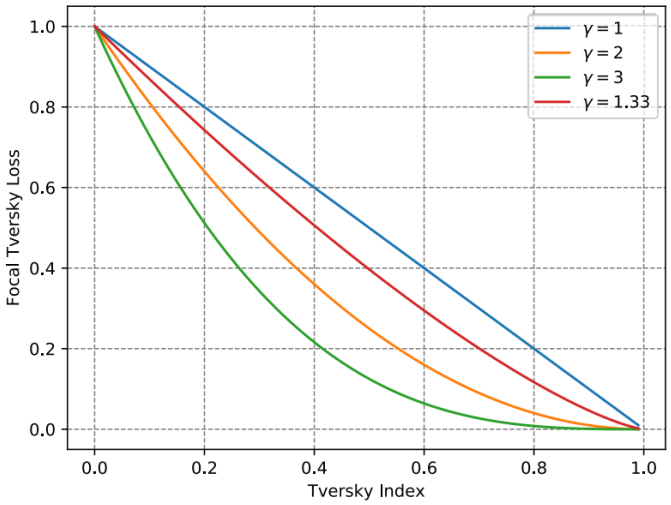


Figure 2.1.4 The focal Tversky loss non-linearly focuses training on hard examples (where Tversky Index < 0.5) and suppresses easy examples from contributing to the loss function (Image Credit: Abraham et al., 2019)

Aside from loss functions, there are different optimizers designed to serve different purposes. In this study, some of the famous optimizers are introduced: Stochastic Gradient Descent (SGD) (Robbins & Monro, 1951), Adaptive Moment Estimation (ADAM) (Kingma & Ba, 2015), AMSGRAD (Reddi et al., 2018), and Adaptive Gradient Methods with Dynamic Bound of Learning Rate (ADABOUND) (Luo et al., 2019).

SGD (Robbins & Monro, 1951) is one of the most dominant algorithms that perform well across many applications. Due to its rapid

training time and reasonable outcome, it is considered as the default optimizer in many networks. However, its characteristic of scaling the gradients uniformly in all directions potentially leads to poor performance. Also, the training speed is limited when the training data points are scattered. To address this issue, recent works have proposed a variety of adaptive methods that scale the gradients by square roots and averaging the squared values of past gradients. As a result, an adaptive optimization method called ADAM (Kingma & Ba, 2015) is proposed to achieve a rapid training process with an element-wise scaling term on learning rates.

Among the adaptive optimizers, ADAM has become one of the most popular optimizers across many deep learning frameworks due to its rapid training speed (Wilson et al., 2017). From (2.1.11), an exponentially decaying average of past squared gradients m_t is computed, and from (2.1.12), an exponentially decaying average of past squared gradients v_t is computed. Then, the average of the gradients is multiplied by the initial learning rate η, and divided by square root of the exponential average of squared gradients in (2.1.13). Finally, in (2.1.14), the weight change $\Delta\omega_t$, is added to update the weight ω_t. Moreover, the authors of ADAM (Kingma & Ba, 2015) proposes the hyperparameters β_1, β_2, and ϵ to be 0.9, 0.999 and 10^{-8}, respectively.

$$m_t = \beta_1 * v_{t-1} - (1 - \beta_1) * g_t \quad (2.1.11)$$

$$v_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2 \quad (2.1.12)$$

$$\Delta\omega_t = -\eta \left(\frac{m_t}{\sqrt{v_t} + \epsilon} \right) * g_t \quad (2.1.13)$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t \quad (2.1.14)$$

Adaptive methods often display faster progress in the initial portion of the training, but their performance quickly plateau on unseen data such as test dataset. Since the model performance heavily depends on the type of the dataset, non-adaptive optimizers such as SGD performs better in some cases. For instance, SGD performs better than adaptive methods in natural language processing and certain computer vision applications (Luo et al., 2019; Wu & He, 2018).

For further improvement, a variant of ADAM called AMSGRAD is proposed by Reddi et al. (2018) to overcome the issue of the generalization ability and out of sample behavior of these adaptive methods. However, AMSGRAD shows better performance on training data, but the generalization ability on test data is found to be similar to ADAM. Luo et al. (2019) report that both extremely large and small learning rates exist by the end of training on ADAM, which indicates that the lack of generalization performance of adaptive methods stemmed from unstable and extreme learning rates. To

remediate this issue, AMSGRAD is introduced with non-increasing learning rates, which helps to abate the impact of huge learning rates. Though, neglecting the possible effects of small learning rates may lead to undesirable non-convergence.

Moreover, another variant of ADAM, named ADABOUND (Luo et al., 2019) is proposed. Unlike ADAM, ADABOUND does not suffer from the negative impact of extreme learning rates. It employs dynamic bounds on learning rates in ADAM, where the lower and upper bounds are initialized as zero and infinity respectively, and eventually converges to constant final step size. Furthermore, ADABOUND is regarded as an adaptive method at the beginning of training and it gradually and smoothly transforms to SGD as the time step increases. By having both aspects from SGD and an adaptive method, the advantage of a rapid initial training process and good final generalization ability is taken (Luo et al., 2019).

2.2 Convolutional Neural Network (CNN)

In the early 1960s, Hubel et al. (1963) discover the concept behind the locally sensitive, and orientation-selective neurons in the cat's visual system and realize that small regions of cells from the visual cortex are sensitive to

specific regions of the visual field. Furthermore, certain individual neuronal cells in the brain respond only in the presence of edges of a certain orientation. For instance, certain neurons respond when exposed to edges with particular orientations. Hubel et al. (1963) notice that these neurons are organized to produce visual perception. The fact that neuronal cells in the visual cortex look for specific characteristics, the fundamental basis behind the neural network are formed from this concept. Thus, local connections have been reused many times in neural models of visual learning (LeCun et al., 1995). With the support of local receptive fields, neurons can extract elementary visual features such as oriented edges, endpoints, corners, etc. Additionally, these receptive fields are applied by forcing a set of units at different places on the image to have identical weight vectors (Rumelhart et al., 1986). Thence, these outputs of such a set of neurons are known as a "feature map". Depending on the corresponding receptive fields, different types of feature maps are collected. This operation is well known as "convolution" and the receptive field is referred to as the "kernel" (LeCun et al., 1995). A convolution layer produces multiple feature maps at each location with different weight vectors. These extracted features prioritize their approximate position relative to other features than their original location. Additionally, pooling layers can be applied after a convolutional layer to reduce the

resolution of the feature map and reduce the sensitivity of the output to shifts and distortions (LeCun et al., 1995).

In the field of machine learning, CNN is the most trending image classification technique to classify objects in an image. CNN is mainly composed of the input layer, hidden layers, and output layer, where the hidden layers are usually composed of convolution layers and pooling layers. In the past few years, the capabilities of CNN have proved its ability to work with all kinds of different image classification tasks. Ever since CNN became so popular, various versions of CNN are developed to handle more complicated tasks than a simple classification. For instance, Fan et al. (2018) designed a CNN based architecture that successfully segments cracks from a pavement image. This network is called the CNN with structured prediction. Unlike a typical CNN, the CNN with structured prediction performs pixel-level classification, which results in a binary segmentation map of the pavement image. This network consists of 4 convolution layers with 2 max-pooling layers, followed by 3 fully connected layers (Fan et al., 2018). Every convolutional layer is applied with a 3x3 kernel and a stride of 1 pixel. Additionally, zero paddings are applied to the boundary of each input image before the convolution filters are applied to preserve the spatial resolution of the feature map. After each pair of convolutional layers, max-pooling is

applied with a stride of 2 over 2x2 kernel. Towards the end of the network, two consecutive fully connected layers with 64 neurons are used, followed by another fully connected layer with 25 neurons. The final outputs of 25 neurons are then reshaped into a 5x5 binary prediction, 1 being a crack pixel, and 0 being a non-crack pixel (Fan et al., 2018). An overview of the network is described in Figure 2.2.1.

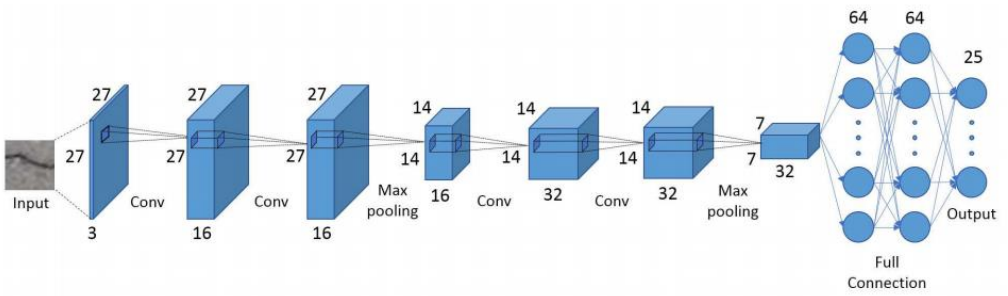


Figure 2.2.1 An illustration of the original CNN with a structured prediction (Image Credit: Fan et al., 2018)

Alternatively, Fan et al. explore different output sizes, ranging from one single pixel, which is essentially identical to the traditional CNN output, and a 7x7 output patch. Using a single pixel results extensive prediction time and a 7x7 pixels results in coarse output. Upon experimenting with different output sizes, Fan et al. report that the output with a 5x5 image patch achieves the best result. Additionally, in a natural pavement image, there are far more non-crack pixels

than crack pixels in a pavement image. Therefore, balancing the data size between these two classes is fundamentally correlated to the final accuracy. Hence, the effect of ratio between the numbers of crack and non-crack image patches is an important factor. Consequently, Fan et al. conclude that a 1:3 ratio of crack to non-crack patches results in the best outcome.

Even though Fan et al.'s work achieves a good precision of ~91%, this network is not able to either determine the severity of cracks or identify the type of cracks. In practice, this additional information is known to be essential in prioritizing repair schedules. Additionally, the output from this network is restricted to tiny image patches that require post-processing of stitching these predicted patches to reconstruct an image. Thus, in this research, a variant of CNN with a structured prediction network was implemented to add a crack severity classification.

2.3 Fully Convolutional Network (FCN)

In general, a typical CNN ends with a fully connected layer to employ the extracted features from the convolution layers and classify the input image into different classes. The biggest advantage of having fully connected layers is that these layers learn features from all the combinations of the features of

the previous layer. However, not only fully connected layers are computationally expensive, spatial sizes are restricted to the dimension of the input. Thus, Long et al. (2015) propose a network called a fully convolutional network (FCN) that is trained end-to-end, pixels-to-pixels on semantic segmentation. Semantic segmentation is mainly divided into global information, which deals with classifying the object, and local information deals with localizing the object in the image. Deep feature hierarchies encode location and semantics in a non-linear local to a global pyramid. Consequently, this approach does not require any pre-processing or post-processing complications.

As mentioned earlier, convolutional networks' basic components include convolution and pooling layers. These operations work on local input regions and depend on relative spatial coordinates. Thus, CNN and FCN both take advantage of convolution layers to extract features, and these convolutional networks are built on translation invariance. Due to the fully connected layers at the end of these networks, the input dimensions are fixed, and spatial coordinates are lost along the way. Typical classification neural networks including LeNet (LeCun et al., 1989), and AlexNet (Krizhevsky et al., 2012) feed the input with a fixed size and produce non-spatial outputs. Although these fully connected layers disregard the spatial coordinates, they

are viewed as convolutions with kernels that cover their entire input regions. Hence, as shown in Figure 2.3.1, these layers are transformed into convolution layers to enable a simple classification network to output a heatmap.

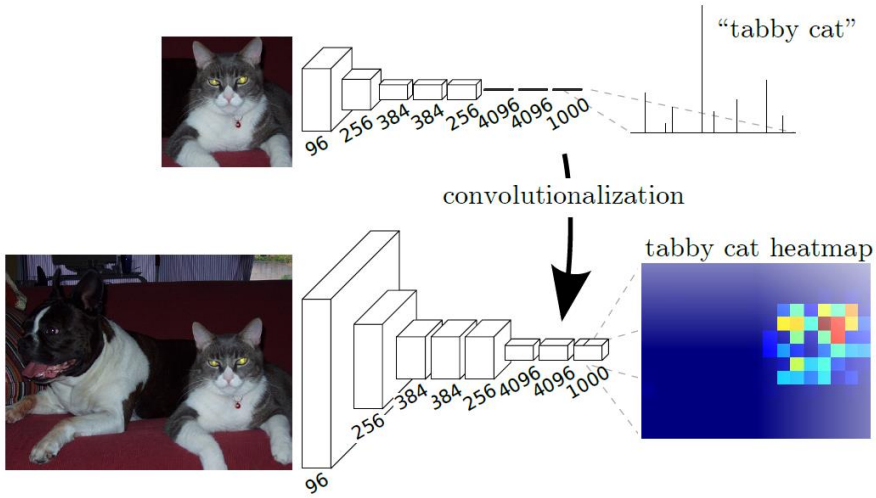


Figure 2.3.1 Transforming fully connected layers into convolution layers enabled a classification net to output a heatmap. Adding layers and a spatial loss produced an efficient machine for end-to-end dense learning (Image Credit: Long et al., 2015)

The spatial output maps of these convolutionalized models make them a natural choice for dense problems like semantic segmentation. However, interpolation is necessary to connect coarse outputs to dense pixels. Hence, Long et al. (2015) utilize backward convolution (a.k.a. deconvolution) to use simple bilinear interpolation to compute each output from the nearest four

inputs by a linear map that depends only on the relative positions of the input and output pixels. Thus, up-sampling is performed within the network to achieve end-to-end learning by backpropagation from pixel-wise loss. Moreover, these deconvolution filters along with its activation functions are capable of learning a non-linear up-sampling as well.

To take advantage of the feature spectrum that combines deep, coarse, semantic information and shallow, fine, appearance information, FCN is proposed with a skip architecture (Long et al., 2015). The summary of skip connections is shown in Figure 2.3.2. As shown in Figure 2.3.2, adding skip connections between layers to fuse coarse, semantic, and local, appearance information is one of the highlights from FCN. Skip architecture learns end-to-end to refine the semantics and spatial precision of the output.

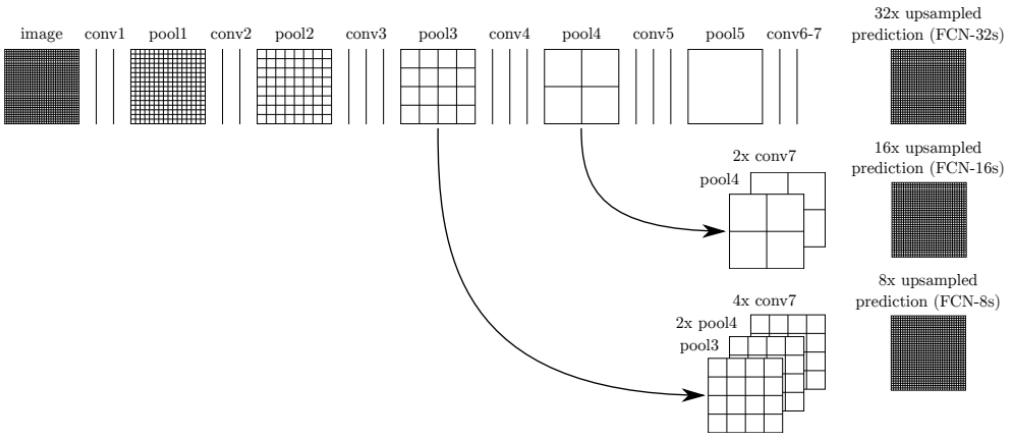


Figure 2.3.2 FCN combining coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s). Second row (FCN-16s): Combining predictions from both the final layer and the pool4 layer, at stride 16, lets our net predict finer details while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from pool3, at stride 8, provide further precision (Credit: Long et al., 2015)

Multiple proven classification architectures including AlexNet (Krizhevsky et al., 2012), VGG net (Simonyan et al., 2014), and GoogLeNet (Szegedy et al., 2015) are convolutionalized as FCN, and Long et al. (2015) claim that fine-tuning from classification to segmentation gave the best result with the VGG-16 network. Without the use of skip connections, the final predictions are very coarse, due to the limitation of 32-pixel stride at the final prediction layer. Thus, combining fine layers and coarse layers make local predictions that consider global structure. To generate FCN-16, output stride is divided in half

to predict using a 16-pixel stride layer. Then, a 1×1 convolution layer is added on top of pool4 to produce additional class predictions. This output is then fused with the predictions computed on top of conv7 at stride 32 by adding a 2× up-sampling layer and these predictions are summed up. Additionally, the up-sampling process is done with the bilinear interpolation. Finally, the stride 16 predictions are up-sampled back to the original image size. This process is visualized in Figure 2.3.2. Similar steps are followed to generate FCN-8, which generates the finest segmentation map. By fusing predictions from pool3 with a 2× up-sampling of predictions fused from pool4 and conv7, FCN-8 is generated. In this fashion, decreasing the stride of pooling layers does result in finer predictions but having smaller stride of pooling layers than FCN-8 is problematic for the VGG-16 network due to its receptive field size and expensive computational cost. With FCN-8, Long et al. achieve 56.0 mean intersection over union on the validation set of PASCAL VOC 2011.

2.4 U-Net

The fundamental structure of the U-Net is similar to the FCN. U-Net uses learnable weight filters instead of fixed bilinear interpolation for up-

sampling. U-Net utilizes transposed convolution to up-sample the feature maps. Also, the skip connections are applied differently in U-Net. Hence, U-Net is known as a modified version of FCN that yields a more precise segmentation map (Ronneberger et al., 2015). The structure of U-Net is shown in Figure 2.4.1, where it is composed of contraction, bottleneck, and extraction section.

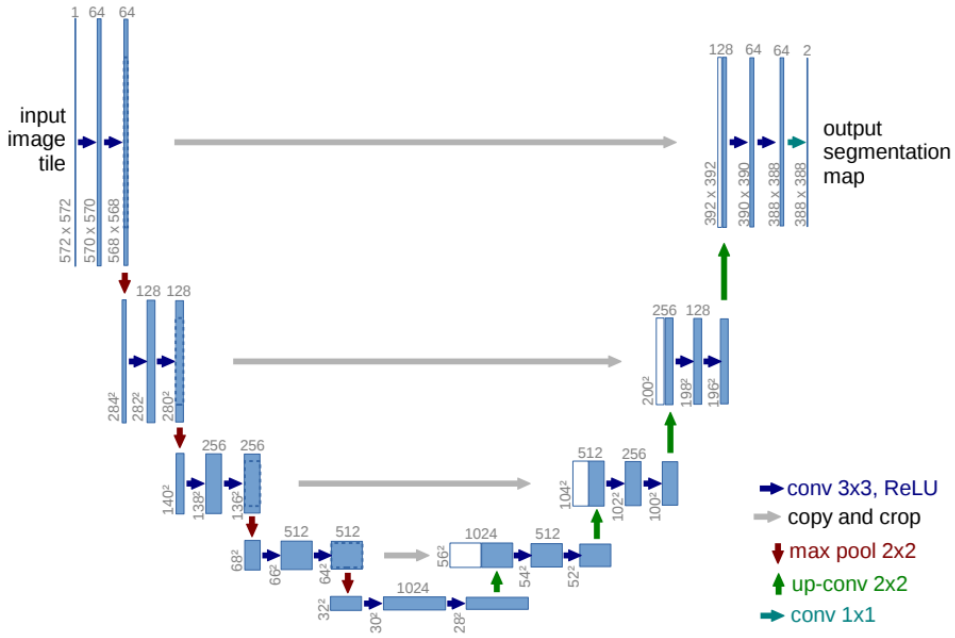


Figure 2.4.1 Original U-Net Architecture (Ronneberger et al., 2015)

The training images are passed to the contracting section of the network by successive layers, and high-resolution features from this contracting section

are combined with the up-sampled outputs. One of the major modifications from FCN is in the up-sampling path. A large number of feature channels in the up-sampling path allow the network to propagate context information to higher resolution layers and consequently, the expansion path becomes symmetric to the contraction path, which yields an u-shaped architecture. U-Net is originally designed to perform cell segmentation, and the biggest challenge in this application is to separate the touching objects of the same class. Hence, Ronneberger et al. (2015) include weights to the loss function by assigning a large weight on the border of background labels and touching cells.

As shown in Figure 2.4.1, the contraction path of U-Net is composed of repeated application of two 3x3 convolution layers, each followed by a ReLU activation function and a 2x2 max pooling operation with stride 2 for down-sampling. The number of convolutional filters is doubled at each down-sampling step. Consequently, the expansion path is composed of 2x2 deconvolution layers, which up-sample the feature maps passed on from the contraction path. Similar to the contraction path, the expansion path also consists of two 3x3 convolution layers, each followed by a ReLU activation function. Additionally, the corresponding feature maps from the contraction path are concatenated to the feature maps at the expansion path. Unlike the contraction path, the number of convolutional filters is halved in the expansion

path. Finally, in the last layer, a 1x1 convolution is employed to map the feature vector to the number of classes to classify.

Additionally, U-Net is trained with an SGD optimizer and the cost function is computed by a pixel-wise softmax over the final feature map combined with the cross-entropy loss function. With this network architecture, the U-Net aims for training on the medical dataset, where there are not many labeled images available as training data. As mentioned earlier, Ronneberger et al. (2015) use a pre-computed weight map for each ground truth to support the network to prioritize learning the ambiguous borders between touching cells. Thus, the weight maps are computed by (2.4.1):

$$w(x) = w_c(x) + w_0 * exp\left(-\frac{\left(d_1(x) + d_2(x)\right)^2}{2\sigma^2}\right) \qquad (2.4.1)$$

where w_c is the weight map to balance the class frequencies, d_1 and d_2 is the distance to the border of the first nearest cell and the second nearest cell, respectively. The initial weight w_0 is set to 10 and σ is set to 5 pixels.

$$E = \sum w(x) \log(p_{l(x)}(x)) \qquad (2.4.2)$$

Moreover, the weight maps computed from (2.4.1) are added to the cross-entropy loss function defined in (2.4.2). As shown in Figure 2.4.2, the ground

truth label is binarized to compute the distance between each cluster of cells and rank them into different weights. More weights are assigned towards cells with smaller gaps in between them and fewer weights are assigned to the cells that have larger gaps between them. Ultimately, U-Net is applied to cell segmentation tasks in light microscopic images from the ISBI cell tracking challenge 2014 and 2015. From these datasets, U-Net achieves an average intersection over the union of 92% and 77.5%, respectively.

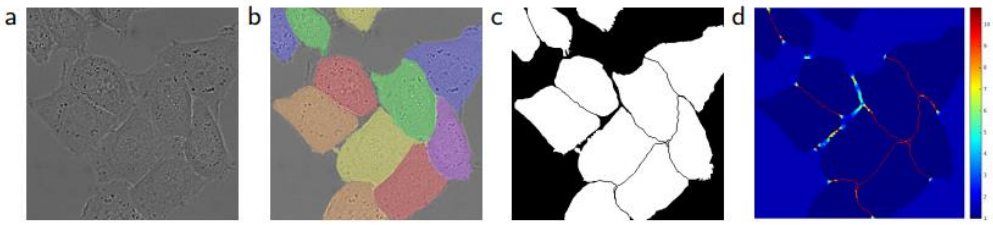


Figure 2.4.2 (a) raw image. (b) overlay with ground truth segmentation. Different colors indicate different instances of the HeLa cells. (c) generated segmentation mask (white: foreground, black: background). (d) map with a pixel-wise loss weight to force the network (Image Credit: Ronneberger et al., 2015)

2.5 Attention U-Net

Despite the good representational power from FCN and U-Net, these architectures rely on multi-stage cascaded CNNs when the target varies in shape and size. These cascaded frameworks are used to extract a specific

region of interest and make corresponding dense predictions on that particular region of interest. However, this cascaded process leads to excessive and redundant use of computational resources and model parameters. For instance, multiple similar low-level features are repeatedly extracted by all models within the cascade. Therefore, Oktay et al. (2018) introduce Attention Gates (AGs) to the U-Net. By employing AGs to a CNN model, the model is trained from scratch in a standard way, but AGs automatically learn to focus on target structures without additional supervision. These gates generate soft region proposals implicitly and highlight salient features useful for the specific task. Most importantly, these gates do not increase significant computational cost. The proposed AGs improve model sensitivity and accuracy for dense label predictions by suppressing feature activations in irrelevant regions. Hence, attention U-Net focuses on image-grid based gating that allows attention coefficients to be specific to local regions. When AGs are implemented to U-Net, the image features are extracted at multiple image scales and coarse feature maps capture contextual information. Then, the category and location of foreground objects are highlighted from these coarse feature maps. These extracted feature maps at multiple scales are merged through skip connections to combine coarse and fine level dense predictions. The overview of the attention U-Net is shown in Figure 2.5.1. As shown in Figure 2.5.1, the feature

maps extracted at the coarse-scale are input through gating to disambiguate irrelevant and noisy responses in skip connections. Then, the output from AG is concatenated to merge relevant activations. In addition, AGs filter the neuron activations during the forward and backward pass. Consequently, gradients originated from background regions are down-weighted during the backward pass. Hence, model parameters in shallower layers are updated based on spatial regions that are relevant to a given task.

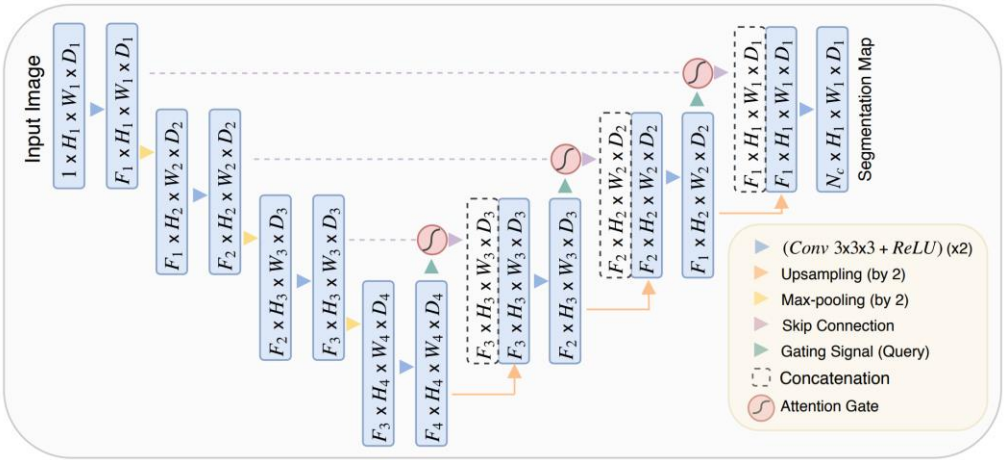


Figure 2.5.1 A block diagram of the proposed Attention U-Net segmentation model. The input image was progressively filtered and down-sampled by a factor of 2 at each scale in the encoding part of the network. N_c represented the number of classes. Attention gates (AGs) filter the features propagated through the skip connections. Feature selectivity in AGs was achieved by the use of contextual information (gating) extracted in coarser scales. (Image Credit: Oktay et al., 2018)

Schematic of the additive attention gate is shown in Figure 2.5.2. As

shown in Figure 2.5.2, Oktay et al. (2018) use additive attention (Bahdanau et al., 2014) to obtain the gating coefficient, over multiplicative attention (Luong et al., 2015). Although additive attention is computationally more expensive, it achieves s higher accuracy. Additive attention is formulated as:

$$q_{att}^{l} = \psi^{T}\left(\sigma_1\left(W_x^{T} x_i^{l} + W_g^{T} g_i + b_g\right)\right) + b_\psi \quad (2.5.1)$$

$$\alpha_i^{l} = \sigma_2\left(q_{att}^{l}\left(x_i^{l}, g_i; \Theta_{att}\right)\right) \quad (2.5.2)$$

where σ_2 indicates the sigmoid function. AG is characterized by a set of parameters Θ_{att}, which consists of a linear transformation and bias term. The linear transformations are computed using 1x1x1 convolutions for the input tensors. Softmax function is typically used in image captioning and classification tasks to normalize the attention coefficients, but sequential use of softmax function yields sparser activations at the output. Hence, the sigmoid function is used instead. These gating signals for each skip connection aggregate information from multiple imaging scales, which increase the grid-resolution of the query signal and achieve optimal results. Ultimately, the attention U-Net becomes the state-of-art for a single model in CT pancreas segmentation, which achieves 81.48 ± 6.23 Dice similarity coefficients for pancreas labels (Oktay et al., 2018).

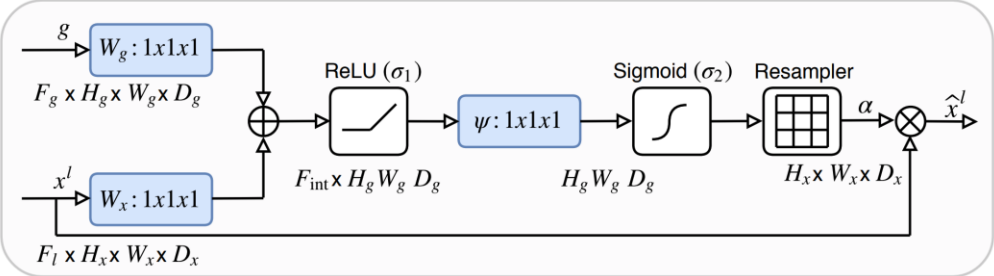


Figure 2.5.2 Schematic of the proposed additive attention gate (AG). Input features (x^l) were scaled with attention coefficients (α) computed in AG. Spatial regions were selected by analyzing both the activations and contextual information provided by the gating signal (g) which was collected from a coarser scale. Grid resampling of attention coefficients was done using trilinear interpolation. (Image Credit: Oktay et al., 2018)

Chapter Three: Methodology

3.1 Dataset

At MTO, the pavement data are collected by Automatic Road Analyzer (ARAN) 9000 system developed by Fugro Roadware (Mississauga, ON, Canada). ARAN 9000 is an advanced pavement data collection vehicle equipped with precision survey systems including video cameras, optical sensors, laser line projectors, ultrasonic sensors, and accelerometers (Cafiso et al., 2017; Gkovedarou, 2019). Due to its high cost, ARAN 9000 is mainly used to collect data on some of the major highways in Ontario, Canada. During data collection, ARAN 9000 saves data every 10 m of the highway. The collected raw data are processed by two software packages, namely, "Laser Crack Measurement System (LCMS) road-inspect", developed by Pavemetrics (Québec, QC, Canada) and "Vision", developed by Fugro Roadware.

Since these software packages utilize the raw data from ARAN 9000, the depth measurements are also taken into consideration when detecting the cracks, but the width of the cracks are mainly used to classify the cracks. Based on the standard of MTO, the different levels of severity of cracks are defined as follows: longitudinal wheel-track cracks with less than 3mm are

classified as very slight level; cracks from 3mm to 12mm are considered slight level; single cracks with 13mm to 19mm width or multiple cracks starting is categorized as moderate level; cracks with 20-25mm width for single cracks or multiple cracks, beginning of spalling are deemed as severe level; and cracks greater than 25mm wide for single cracks or multiple cracks with developed spalling are considered very severe level (MTO, 2016).

Based on these standards, LCMS road-inspect and Vision software detect the cracks from the laser data collected by ARAN 9000. Although ARAN 9000 comes with video cameras, these cameras are facing forward and sideways to record the surrounding environments rather than facing down on the pavement to capture the pavement conditions. Therefore, the range of the laser data was saved as single-channel images through LCMS road-inspect software and these images were used as the training images in this study. A sample of the pavement image is shown in Figure 3.1.1.

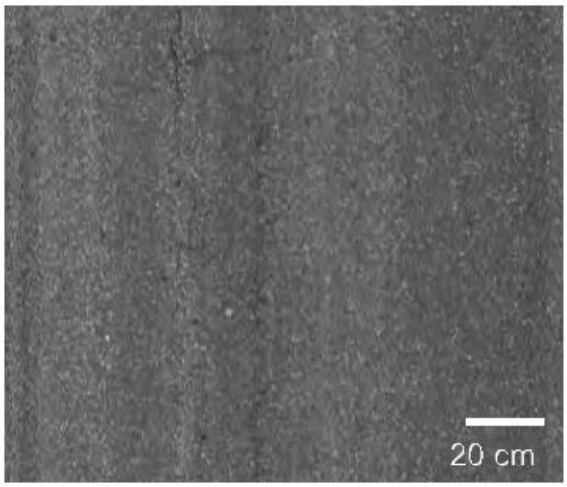
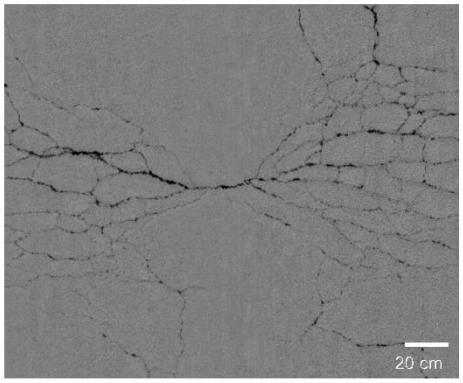


Figure 3.1.1 Sample pavement image

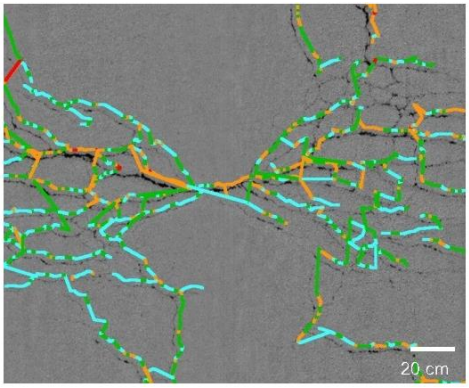
Even though both the LCMS road-inspect and Vision software package was capable of processing and analyzing the raw data from ARAN 9000, they used different criteria to detect and determine the severity of the distress levels on the pavement. In general, the LCMS road-inspect software had a lower localization accuracy than the Vision software. LCMS road-inspect algorithm was set up with a lenient threshold. Hence, there were false positive cracks detected along with the true positive cracks. Meanwhile, the Vision software was set up with a stricter threshold than the LCMS road-inspect software. Hence, it yielded a relatively fewer number of false positive cracks detected than the LCMS road-inspect software. However, a stricter threshold resulted in misclassifying some of the small cracks as non-cracks. Moreover, the Vision

software often had trouble distinguishing between the low and medium severity cracks. Nonetheless, its outcome was more precise in localizing the cracks on the pavement than the outcome from the LCMS road-inspect software. Consequently, the resulted labels from these two software packages were not necessarily identical. Although the Vision software localized the cracks better than the LCMS road-inspect software, it did not provide accurate severity labels. Undoubtedly, the cracks at the higher severity levels were easy to be detected by both software packages, but there were discrepancies in the detection of cracks at the low and medium severity levels. As an example, the difference in the resulted labels between the two software packages is shown in Figure 3.1.2.

(a)



(b)



(c)

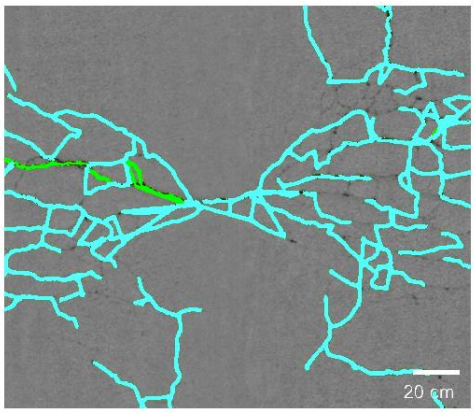


Figure 3.1.2 Results obtained by LCMS and Vision: (a) Original pavement image, (b) cracks detected by LCMS, and (c) cracks detected by Vision (Low: Cyan, Medium: Green, High: Orange)

To generate the best ground truth, the labels created from both software packages were merged to take advantage of the accurate crack localization with properly labeled severity levels. Thus, only crack labels that appeared on both software packages were used as the true positive crack pixels and their corresponding severity levels were adapted after the LCMS road-inspect software. It is worth mentioning that even though the labeled images from two different software packages were merged, it still did not fully capture all the cracks on the pavement images. In this study, a total of 3,332 pavement images with corresponding merged ground truth images were used to train and test the convolution-based neural networks. Each image represented a 10 m segment of the highway and had a dimension of 2500x1037x1 pixels.

For data augmentation, the images were subdivided into 160x578 pixels with an overlap of 75%. Since the ground truth labels were imperfect, the road lines were often misclassified as cracks. Therefore, when the images were subdivided into smaller images, a region of interest was set to collect the images within the road lines. As a result, ~65,000 images were available for training and validation, with additional ~400 images for testing purposes. Hence, ~65,000 images were randomly split into 70% of training data and 30% of validation data to train the neural networks in this research. Then, ~400

unseen test images were used to evaluate the performance of the models.

3.2 Improved CNN with A Structured Prediction

Fundamentally, CNN with a structured prediction proposed by Fan et al. (2018) had its basis on a typical CNN; the only difference was that the network resulted in a patch-wise prediction instead of a single prediction. Similar to a typical CNN architecture, CNN with a structured prediction proposed by Fan et al. (2018) was composed of an input layer, hidden layers, and output layer. Using this network, Fan et al. (2018) successfully predicted a binary map of cracks on paved roads. Thus, CNN with a structure prediction proposed by Fan et al. (2018) was improved in the following aspects in this study:

(1) CNN with a structured prediction proposed by Fan et al. (2018) was adapted to classify cracks with different levels of severity.
(2) Weighted loss function was proposed to effectively train the network.
(3) ADABOUND optimizer was employed instead of the ADAM optimizer used in Fan's network.

In order to adapt the CNN with a structured prediction proposed by Fan et al. (2018) to classify cracks with different levels of severity, the overall number of layers in the network was increased to extract more features, and the number of output units in the output layer was increased. Hence, by preserving the same output size of 5x5 from the original Fan's network, the network proposed in this study required 25 neurons per class, resulting in 75 neurons in the final output layer. The architecture of the improved CNN with a structured prediction proposed in this study was shown in Figure 3.2.1. As shown in Figure 3.2.1, the number of convolutional layers was increased to 6; the number of max-pooling layers was increased to 3, and the number of fully connected layers remained the same but the number of neurons within each fully connected layer was increased. All convolutional layers were equipped with kernels of 3x3 and stride of 1. Additionally, each convolutional layer was applied with zero paddings on the boundary to preserve the spatial resolution of the resulting feature map. Moreover, max-pooling with kernels of 2x2 and stride of 2 was applied on the feature map after the convolutions. Since the expected output can have multiple identical labels with different classes, this structured prediction problem was modeled as a multi-label, multi-class problem.

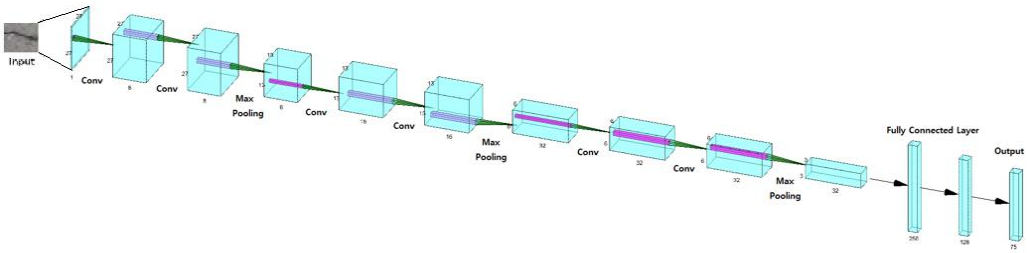


Figure 3.2.1 Modified CNN with Structured Prediction Architecture

Within each original training image, approximately 300 image patches of size 27x27 were extracted. The corresponding label patches of 5x5 were also collected based on the center location of the image patch. To sustain the equal ratio of patches collected over different severity levels, an equal number of patches were collected for each severity level. Moreover, twice the number of non-crack patches were collected than the number of crack patches to simulate the natural ratio of crack and non-crack pixels on a pavement image. In a natural pavement image, there are far more non-crack pixels than crack pixels. Hence, the proportion of positive and negative samples has a huge impact on the performance of the network. To be fair, all samples were randomly extracted from the image, with the ratio of positive to negative samples set to 1:2. As shown in Table 3.2.1, the positive samples consisted of approximately 0.33 ratio of each severity level. In total, 398,925 training

samples were used to train the network. A common practice to optimize the input image was normalizing the input pixel range of [0, 255] to [0, 1], but this pre-processing procedure was not applied directly to the input data. Instead, this procedure was included at the beginning of the network to avoid extra pre-processing procedures outside the network.

Table 3.2.1 CNN data structure

	Positive Samples			Negative Samples
	Low	Medium	High	Non-Crack
Number of Samples	44,325	44,325	44,325	265,950

Since this network's prediction was multi-label and multi-class output, every output unit was not mutually exclusive, and multiple positive predictions were possible. Therefore, weighted binary cross-entropy was used as the loss function, which is defined in (3.2.1):

$$L = -\sum_{i=1}^{s} w * (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)) \qquad (3.2.1)$$

Where y_i and $\hat{y}_i$ is the label and prediction of the ith output unit, respectively; w indicates weight for each corresponding location of the pixels; s represents the number of output units. On top of a regular binary cross-entropy loss function that Fan et al. employed, the weights w was added to focus on the center of the image patch. The nature of structured prediction had its basis on

the central pixels. For instance, if the pixel at the center of a window was a crack pixel, it was very likely for the pixels around it to be crack pixels. The pixels that were directly touching the pixel at the center of a window have a much higher correlation than the pixels that were not in direct contact with the pixel at the center of a window. The level of correlations with respect to the center pixel is shown in Figure 3.2.2.

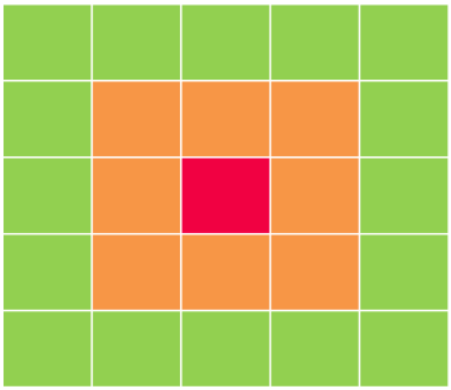


Figure 3.2.2 5x5 Output Window (outermost pixels have a smaller correlation than the inner pixels)

By proposing weights based on the location of the pixels, the prediction accuracy was improved. During the initial experiments, the weights were assigned by the multivariate normal distribution. Specifically, bivariate normal distribution was employed instead of the univariate normal distribution, because the weights were given for both row and column-wise. By varying the

mean value and the standard deviation value, the overall shape of the distribution altered. However, the mean value was fixed to zero for this experiment, because the distribution had to be centered at zero for consistency. Thus, the change in standard deviation values was tested. Upon testing with a large standard deviation value, the resulting weights were more fairly distributed rather than having a high weight at the center pixel, but when a small standard deviation value was used, more weights were given towards the center pixel than the surrounding pixels. This experiment was held with a standard deviation value of 0.25, 0.5, and 0.75. As expected, assigning different weights on the loss function changed the resulting prediction accuracy. The effect of different weights on the loss function is further discussed in the “Discussion”.

Since the network was designed to input a 27x27 image patch, the test images had to be subdivided into small image patches. Hence, the overlaps between image patches were tested, and the decision within the overlapping prediction pixels was ambiguous and did not improve the results either. Additionally, the prediction time of a test image was greatly affected by the overlaps between the image patches. Having a large overlap between the image patches resulted in more number of input image patches to cover the whole test image. Hence, the input patches were generated every 5 pixels.

Hence, there were overlaps between the input patches but not on the predicted output patches. The final prediction of 75 output units was reconstructed into a dimension of width, length, and depth. Depth was composed of three severity levels: low, medium, and high severity. Since the sigmoid activation function was used to compute the probability, each output unit ranged from 0 to 1. Then, the decision probability was set to 0.5 to disregard the pixels with low probability as crack pixels. Finally, the predicted segmentation map was color-coded depending on the severity level: low severity was represented by cyan color; medium severity was represented by green color; high severity was represented by orange color.

3.3 Implementation of FCN, U-Net & Attention U-Net in crack detection

In this study, the following neural networks including FCN (Long et al., 2015), U-Net (Ronneberger et al., 2015), and Attention U-Net (Oktay et al., 2018) were re-designed to correctly segment the cracks from the pavement images and classify the segmented cracks into corresponding severity levels. Fundamentally, FCN, U-Net, and attention U-Net architectures utilized skip connections to combine spatial information from the down-sampling path and

the up-sampling path, but these skip connections were applied differently depending on the network. FCN utilized the addition operator to apply the skip connections to the corresponding layers in the up-sampling path, whereas U-Net employed a concatenator operator. While FCN and U-Net were proven to work well in segmentation tasks, they accompanied many redundant low-level feature extractions, despite that the feature representation is poor in the initial layers. Hence, attention U-Net was introduced with Attention Gates (AGs). AGs were used to only highlight relevant activations during the training phase. The additive soft attention used in attention U-Net was implemented at the skip connections to suppress activations in irrelevant regions and reduce the number of redundant features. Detail of these network's backgrounds was discussed in Chapter 2.

Unlike the input from CNN with structured prediction, FCN and U-Net were capable of employing the whole image without subsampling small image patches. In total, ~65,000 images of size 160x578 pixels were used to train and validate the network. Prior to the training phase, the dataset was randomly split into 70% of training data and 30% of validation data. Furthermore, an overlap of 75% was allowed between the extracted images to increase the number of training images. While pre-processing the input pixel range of [0, 255] to [0, 1] was a common practice to optimize the image in machine

learning, this normalization procedure was not applied directly to the input data. Instead, this normalization was included within the network to skip this procedure when testing on unseen data.

The weights and model structure from VGG-16 was imported to construct FCN for pavement crack detection. Since the pre-existing model structure was used to construct FCN, there were not too many settings to be altered for experiments. Among different FCN architectures, FCN-8 was employed, because Long et al. (2015) verified that it produces the finest segmentation map. Considering that FCN-8 was designed to surpass FCN-16 and FCN-32, only FCN-8 was exploited in this research. The overview of the FCN architecture was shown in Figure 3.3.1. Figure 3.3.1 described which layers were combined to construct FCN-32, FCN-16, and FCN-8. In this architecture, the number of outputs was set to 4 classes: no crack, crack with low severity, medium severity, and high severity.

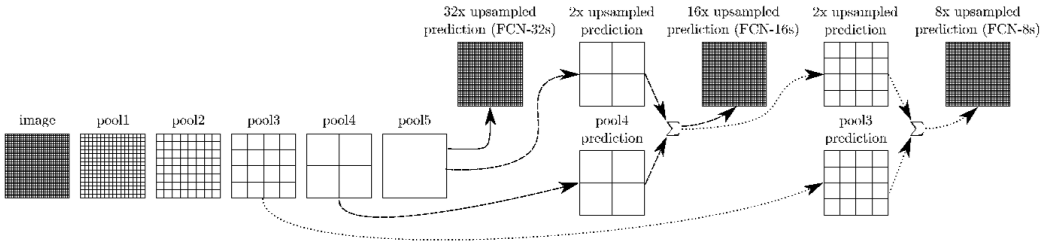


Figure 3.3.1 VGG-16 based FCN Architecture

Segmentation maps from FCN-32 were directly produced from the last convolution layer by using a transposed convolution layer with a stride of 32. However, the resulting segmentation map from FCN-32 was too coarse. Hence, the finer version of the segmentation maps was produced by FCN-16. The up-sampled prediction from the last convolution layer was summed up with the prediction from pool4 shown in Figure 3.3.1, and by using a transposed convolution layer with a stride of 16, FCN-16 was constructed. Similarly, the result from the up-sampled version of the last convolution layer was summed up with pool4 shown in Figure 3.3.1, and this result was once again up-sampled then summed up with pool3 to produce the FCN-8 by using a transposed convolution layer with stride 8.

In order to properly adapt the U-Net for pavement crack application, U-Net architecture was re-designed in this study, and the architecture is shown in Figure 3.3.2.

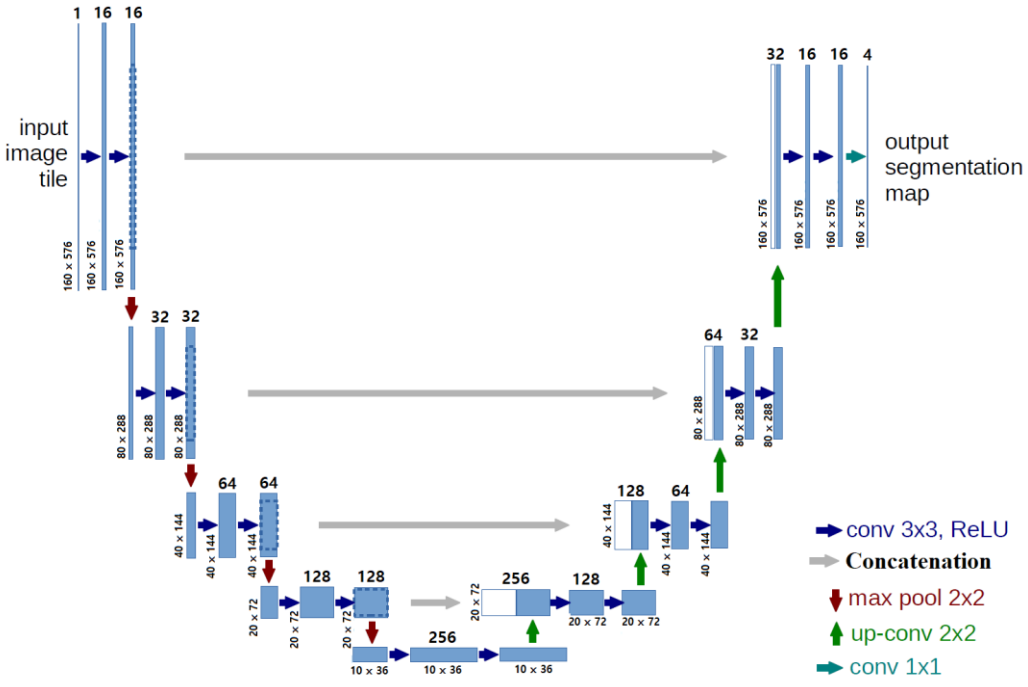


Figure 3.3.2 U-Net Architecture

The major change from the original U-Net (Ronneberger et al., 2015) was reducing the number of convolution filters by a factor of 4. Since the pavement cracks required mostly low-level features such as lines, edges, and curves, having too many convolution filters was unnecessary. Also, the model with a reduced number of convolution filters trained and predicted faster than the original U-Net architecture. The input and output of the image were identical to FCN. Unlike FCN, the skip connections were applied on to the

expansion path by concatenation operator, instead of the addition operator. The original U-Net used an SGD optimizer with a cross-entropy loss function, but in this study, ADAM optimizer and ADABOUND optimizer with cross-entropy and focal Tversky loss function were investigated. The detailed analysis of these experiments was discussed in "Discussion".

Furthermore, attention U-Net was explored to reduce the effect of class imbalance. As shown in Figure 3.3.3, the attention U-Net was designed with the same number of neurons and layers as U-Net described in Figure 3.3.2.

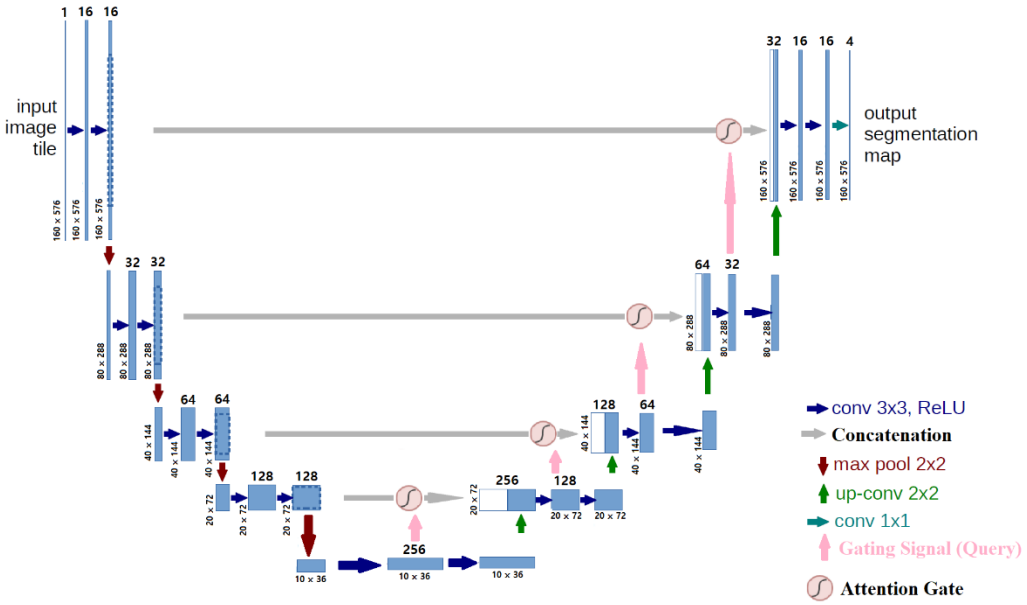


Figure 3.3.3 Attention U-Net Architecture

The major difference from U-Net architecture was the addition of attention gates before the concatenation operation. Similar to medical applications, the images from the pavement crack application consisted of a majority of negative pixels (non-crack pixels) and a small portion of positive pixels (crack pixels). Essentially, when the feature maps from the down-sampling path and up-sampling path were combined, the aligning weights were amplified while unaligned weights became relatively smaller. In the case of crack applications, the extracted features of the crack segments were highlighted with higher weights, and the surrounding non-crack regions were trained with lower weights. By prioritizing the salient features, attention U-Net was trained more efficiently than U-Net. The attention gate and the gating signal was implemented the same way as the original attention U-Net proposed by Oktay et al. (2019). In addition, focal Tversky loss was experimented on this attention U-Net to further filter out the unnecessary portion of the pavement image (Abraham et al., 2019). Moreover, the attention U-Net was explored with both ADAM and ADABOUND optimizer.

FCN, U-Net, and attention U-Net were trained for 50 epochs with a batch size of 23 training images per batch. Due to the limitation of GPU memory, 23 training images were the maximum number of images per each

batch. Unlike the CNN with a structured prediction approach, FCN, U-Net, and attention U-Net did not require any image stitching or post-processing after the prediction. Once the input test images were predicted through these approaches, predicted images were evaluated in the same fashion as the CNN approach. Identical test images were used to be consistent with the evaluation results.

3.4 Performance Evaluation

The network was trained with Intel core i7-8700 3.20GHz CPU, 16GB RAM, and Nvidia GeForce RTX 2070 GPU. The code was written under python with Tensorflow and Keras. The performance of the network was determined by the test results. For evaluations, precision, recall, and F1-score were used for classification tasks, as shown in (3.4.1), (3.4.2), and (3.4.3):

$$Precision = \frac{TP}{TP + FP} \quad (3.4.1)$$

$$Recall = \frac{TP}{TP + FN} \quad (3.4.2)$$

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (3.4.3)$$

where TP, FP, FN are the number of true positive, false positive, and false

negative, respectively. Due to the representation of different severity levels, the ground truth was not generated manually. As mentioned earlier, the labels used in this research did not perfectly detect all the cracks or in worst cases, cracks were not detected at all. Most importantly, the detected cracks were not labeled with respect to the actual width of the cracks. Considering that the crack labels had an equal size of width regardless of the actual width of the cracks, this became a problem when evaluating the results on a pixel-level basis. Due to the nature of these labels, thin cracks ended up having labels that were larger than the actual width and thick cracks ended up having labels that were smaller than the actual width. Thus, when the prediction map was evaluated, numerous true positives were then considered as false positive. In order to minimize this effect, the 5-pixel distance between the predicted label and the true label was set as the threshold. Besides, the transitional areas between crack pixels and non-crack pixels were ambiguous in some cases. Hence, a minimum of 2 pixels or 5 pixels of the threshold in crack detection application was necessary (Amhaz et al., 2016; Fan et al., 2018).

Chapter Four: Results

4.1 Results from different networks

A couple of predicted images from improved CNN with structured prediction, FCN, U-Net, and attention U-Net methods are shown in Figure 4.1.1 as examples.

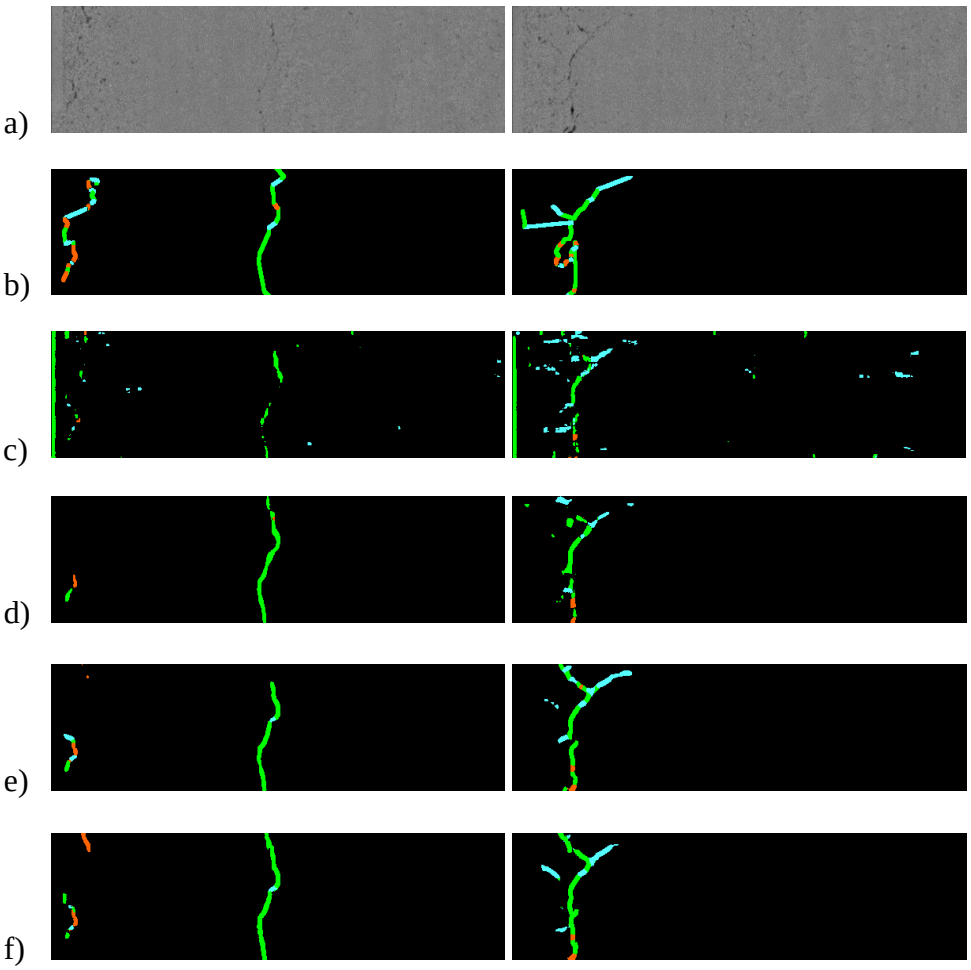


Figure 4.1.1(a) Pavement Image, (b) Given Ground Truth from MTO, (c) Prediction from CNN with ADABOUND optimizer and weight (5, 3, 1) (d) Prediction from FCN with ADAM optimizer (e) Prediction from U-Net with ADABOUND optimizer and Focal Tversky loss (f) Prediction from Attention U-Net with ADABOUND optimizer and Focal Tversky loss (Low: Cyan, Medium: Green, High: Orange)

Amongst all the approaches attempted in this study, the improved CNN with a structured prediction resulted in the most number of false positive pixels. As shown in the left column of Figure 4.1.1, the crack segment located in the middle of the image was extremely thin, and the improved CNN with a structured prediction did not fully detect such a thin crack segment. Although it captured some portion of the crack segment, the detected crack segments had discontinuities in between, whereas other approaches successfully capture the entire crack segment properly. Additionally, the improved CNN with a structured prediction failed to detect the thin cracks on the left side of the image. Furthermore, the improved CNN with a structured prediction resulted in many false positive pixels around the alligator crack shown in the right column of Figure 4.1.1. However, it did capture the overall shape of the crack on the image.

Generally, FCN performed better than the improved CNN with a structured prediction approach. FCN also resulted in some false positive pixels, but it detected low severity cracks much better than the improved CNN with a

structured prediction. Likewise, U-Net showed similar performance as FCN, but it detected the thin cracks slightly better than FCN. Ultimately, the attention U-Net produced the finest-grade prediction result with minimal misclassifications.

Once the networks were trained, the prediction time also varied depending on the network architecture. On average, the improved CNN with a structured prediction took ~80 s, FCN took ~1 s, U-Net, and attention U-Net took ~0.1 s to predict a single image of size 160x576. Undoubtedly, the improved CNN with a structured prediction took the longest time to predict a single image, because it required pre-processing of sub-dividing the input image and post-processing of stitching the predicted patches. In contrast, FCN, U-Net, and attention U-Net did not require any pre-processing or post-processing that slowed down the prediction time. Also, the whole image was directly inputted to these networks and the corresponding prediction map was acquired.

The numerical evaluation results recorded for each network's best setting was shown in Table 4.1.1. Conclusively, attention U-Net showed the highest precision among all the networks attempted. Attention U-Net achieved a higher precision of ~24% than the improved CNN with a structured prediction in low-level severity and ~18% higher precision in medium-level

severity. Not only the attention U-Net was superior on lower level severities, but it also performed better or at least equivalent to other networks on medium and high-level severities. Since attention U-Net was trained with a focal Tversky loss function, which filters out the easy background to focus on the region of interest, it was expected to perform better than other networks. Although these neural networks had an outstanding result, having some of the falsely labeled cracks from the ground truth resulted in lowering the numerical evaluation. Moreover, the falsely labeled cracks often existed in low-level severities because cracks with medium or high-level severities were generally easy to be distinguished. Consequently, cracks with low-level severity resulted in the lowest accuracy than any other severity levels.

Table 4.1.1 Evaluation for CNN, FCN, U-Net and Attention U-Net

	Precision (%)			Recall (%)			F1-Score (%)		
	Low	Med	High	Low	Med	High	Low	Med	High
CNN	12.02	42.28	79.76	86.52	97.44	96.07	19.63	55.60	85.89
FCN	27.50	59.59	84.23	49.08	89.53	98.63	32.08	68.82	89.89
U-Net	34.12	61.31	82.60	61.42	87.62	98.70	40.06	69.97	89.07
Att. U-Net	36.03	60.39	84.58	54.89	94.11	98.14	40.53	71.27	89.95

Chapter Five: Discussions

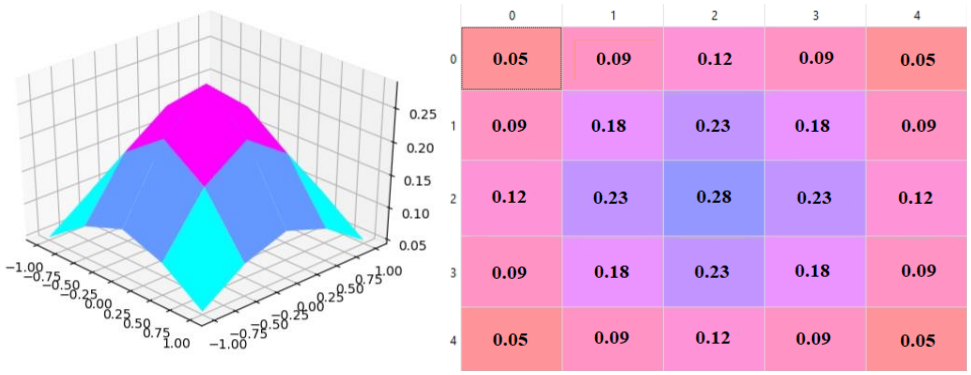
For each network architecture, different settings of the networks were explored to find the optimal configuration that produced the best outcome. Even with the identical network structures, they converged at different stages depending on the optimizer and loss function used. The effects of the loss function and optimizers for each of the networks were investigated in this study.

5.1 The effect of loss functions and optimizers on the improved CNN with a structured prediction

Although CNN with a structured prediction produced a reasonable outcome with a typical cross-entropy loss function and ADAM optimizer (Jung et al., 2019), the strength of spatial correlation between the pixels can be boosted with the weighted cross-entropy loss function and ADABOUND optimizer. As expected, depending on the assigned weights on the loss function, the network resulted in different accuracies. Before assigning fixed values to the weights, bivariate normal distribution was used to set the weights on the loss function. The weights assigned by bivariate normal distribution

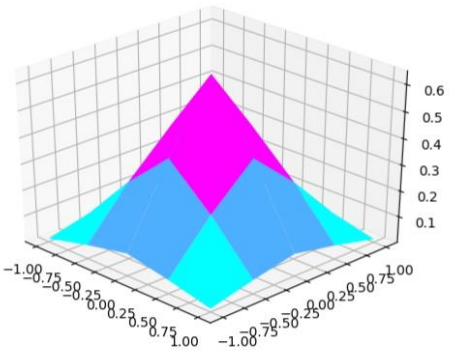
were shown in Figure 5.1.1, Figure 5.1.2, and Figure 5.1.3. In a bivariate normal distribution, smaller standard deviation values resulted in more weight towards the center pixel. However, when the standard deviation was too small, the outermost pixels ended up having extremely small weights. Consequently, the cells located away from the center pixel barely had any contribution to the loss function as their weights were too small. As shown in Figure 5.1.1, and Figure 5.1.2, having a standard deviation of 0.75 and 0.5 allocated the weights more evenly than the standard deviation of 0.25, which ignored the outermost cells.

Along with these weights assigned by the bivariate normal distribution, the network was trained with ADAM and ADABOUND optimizer. Once the network had been trained, the trained network was tested on ~400 test images to evaluate precision, recall, and F1 score.



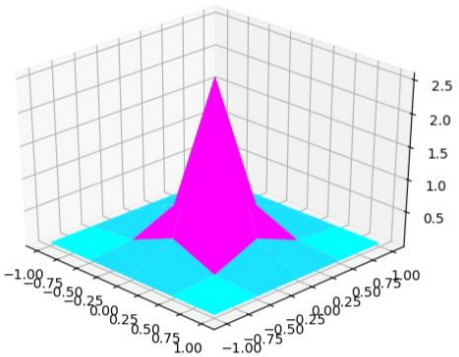
	0	1	2	3	4
0	0.05	0.09	0.12	0.09	0.05
1	0.09	0.18	0.23	0.18	0.09
2	0.12	0.23	0.28	0.23	0.12
3	0.09	0.18	0.23	0.18	0.09
4	0.05	0.09	0.12	0.09	0.05

Figure 5.1.1 Bivariate Normal Distribution with a standard deviation of 0.75



	0	1	2	3	4
0	0.01	0.05	0.09	0.05	0.01
1	0.05	0.23	0.39	0.23	0.05
2	0.09	0.39	0.64	0.39	0.09
3	0.05	0.23	0.39	0.23	0.05
4	0.01	0.05	0.09	0.05	0.01

Figure 5.1.2 Bivariate Normal Distribution with a standard deviation of 0.5



	0	1	2	3	4
0	2.87e-07	1.16e-04	8.54e-04	1.16e-04	2.87e-07
1	1.16e-04	0.05	0.34	0.05	1.16e-04
2	8.54e-04	0.34	2.55	0.34	8.54e-04
3	1.16e-04	0.05	0.34	0.05	1.16e-04
4	2.87e-07	1.16e-04	8.54e-04	1.16e-04	2.87e-07

Figure 5.1.3 Bivariate Normal Distribution with a standard deviation of 0.25

The evaluation results are described in Table 5.1.1 and Table 5.1.2 when trained with ADAM optimizer and ADABOUND optimizer, respectively. When the evaluation results were analyzed, the results on medium and high severity was prioritized rather than low severity, because the crack segments with medium or high severity were the potential danger to the

public. The main purpose of crack detections was to prevent the cracks from developing into a bigger crack by repairing them beforehand. Therefore, it was more meaningful to have higher accuracy in medium and high severity levels than low severity level.

Table 5.1.1 ADAM with bivariate normal distribution weights

ADAM	Precision (%)			Recall (%)			F1-Score (%)		
Weight	Low	Med	High	Low	Med	High	Low	Med	High
$\sigma = 0.25$	8.73	30.92	77.03	94.66	94.83	98.26	15.16	43.48	84.97
$\sigma = 0.5$	10.58	31.34	74.96	96.91	96.05	99.00	18.04	44.13	83.92
$\sigma = 0.75$	11.28	33.25	75.45	95.19	97.40	98.31	19.13	46.64	84.09

Table 5.1.2 ADABOUND with bivariate normal distribution weights

ADABOUND	Precision (%)			Recall (%)			F1-Score (%)		
Weight	Low	Med	High	Low	Med	High	Low	Med	High
$\sigma = 0.25$	3.15	5.06	13.98	99.80	99.86	99.77	6.05	9.53	23.72
$\sigma = 0.5$	6.56	37.27	80.24	87.88	94.22	98.06	11.58	49.21	87.16
$\sigma = 0.75$	7.72	30.79	77.49	88.35	95.67	98.33	13.37	43.22	85.40

As shown in Table 5.1.1, when the network was trained with ADAM optimizer, the overall precision was highest when the weights were assigned by a bivariate normal distribution with a standard deviation of 0.75.

Consequently, the F1 score was also the highest in this case. As the standard deviation was increased for each experiment, both precision and recall improved. However, the highest precision in high severity was achieved when the standard deviation was 0.25, but the difference was only ~1.5% and the standard deviation of 0.75 achieved higher precision in other severity levels. Hence, it was more beneficial to choose the weights assigned by the standard deviation of 0.75, because there was more improvement in percentage-wise for both low and medium severity. Hence, the ~1.5% of deterioration at high severity precision was neglected. Along with the precision, the recall rate improved at a standard deviation of 0.75.

From Table 5.1.2, when the ADABOUND optimizer was used to train the network, precision at all severity levels were very poor when the standard deviation was set to 0.25. In this case, even the precision at high severity only achieved ~14%. When the standard deviation was increased to 0.5, the precision rate dramatically improved from ~5% to ~37% and ~14% to ~80% at medium and high severity, respectively. Unexpectedly, there were no improvements when the standard deviation was increased from 0.5 to 0.75. Unlike the ADAM optimizer, ADAMBOUND achieved the best evaluation results when weights were assigned by the standard deviation of 0.5.

Ultimately, the ADAM optimizer showed the best performance with

bivariate normal distribution when the standard deviation was 0.75 and the ADABOUND optimizer showed the best performance when the standard deviation was 0.5. When these results from both optimizers were compared, ADABOUND optimizer performed worse in low severity but showed better performance at medium and high severity. There was an improvement of approximately 5% of the precision rate for both medium and high severity level. Ideally, more weights were desired at the center than the outer cells. Therefore, assigning fixed weights, including equal weights throughout the cells were attempted to observe the difference from the bivariate normal distribution weights. The results when the network was trained with ADAM optimizer with fixed weights are shown in Table 5.1.3.

Table 5.1.3 ADAM with fixed weights

ADAM	Precision (%)			Recall (%)			F1-Score (%)		
Weight	Low	Med	High	Low	Med	High	Low	Med	High
(1, 1, 1)	9.17	31.19	74.66	90.44	94.37	98.85	15.63	43.33	83.77
(5, 3, 1)	5.84	30.64	76.90	93.50	93.36	97.80	10.55	43.09	84.80

Unexpectedly, as shown from Table 5.1.3 and Table 5.1.4, the network trained with ADABOUND optimizer and fixed weights of (5,3,1) ended up having higher accuracy than the weights produced from the bivariate normal

distribution. The fixed weights were assigned as shown in Figure 5.1.4. Even without setting weights, having equal weights throughout every cell did perform well on the high severity, but not as well on low and medium severity. In terms of accuracy, it was harder to achieve high accuracy as it required both localization and correct severity classification to be considered as a “correctly classified pixel”. Thus, this type of application was more complicated than a simple binary localization of the cracks. Since the ground truth labels were “imperfect”, the transitional areas between different severity levels and the transition from non-crack pixels to crack pixels were very easy to be misclassified. Additionally, the difference in pixel intensity between the crack with low severity and no crack was barely distinguishable. Hence, cracks with low severity had the greatest number of false positives among the three severity levels.

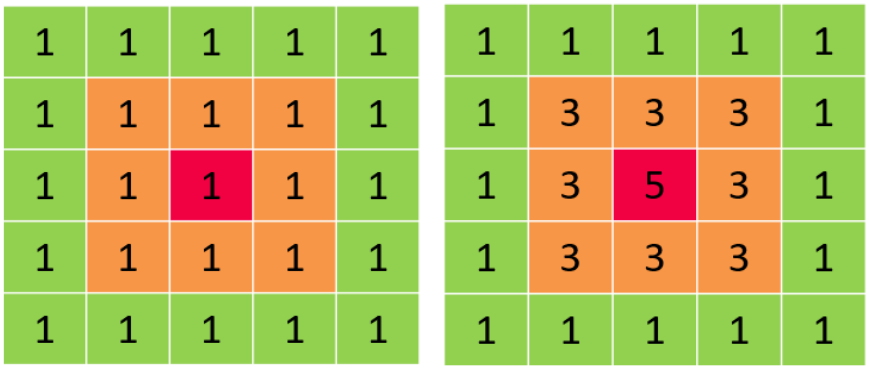


Figure 5.1.4 (1,1,1) Weights (left) and (5,3,1) Weights (right) assigned to loss function

Table 5.1.4 ADABOUND with fixed weights

ADABOUND	Precision (%)			Recall (%)			F1-Score (%)		
Weight	Low	Med	High	Low	Med	High	Low	Med	High
(1, 1, 1)	2.76	20.37	74.67	88.49	90.64	98.56	5.21	30.85	83.51
(5, 3, 1)	12.02	42.28	79.76	86.52	97.44	96.07	19.63	55.60	85.89

From Table 5.1.4 and Figure 5.1.5, the numerical evaluation and sample predictions of the improved CNN with a structured prediction model when it was trained with weights, and without weights were shown. Even when the network was trained without any additional weights, the results were reasonable. However, when the model was trained with weights, the performance in low and medium severity was improved and the prediction results had noticeably less number of misclassified pixels. This indicated that

the performance of the network was boosted by using a weighted loss function because the network not only showed a better prediction performance, it also showed better numerical evaluation results.

The ground truth shown on the left column of Figure 5.1.5 had cracks labeled on the right end of the image but in fact, it was falsely labeled. By learning these wrong features, the network can be confused when predicting similar input pixels. Nonetheless, the network was trained with the majority of correct labels and the wrongly labeled cracks on the right end were not detected as crack pixels. Moreover, the image shown on the right column of Figure 5.1.5 had a dark line along the road lane and this was wrongly classified as a crack segment with high severity. As mentioned earlier, the ground truth used in this study was not manually inspected, and some of the labels were falsely labeled as cracks. Therefore, the network inevitably learned these features as a crack pixel, making the same mistake as the ground truth. This was unfortunately unavoidable when the network was trained with training data with outliers.

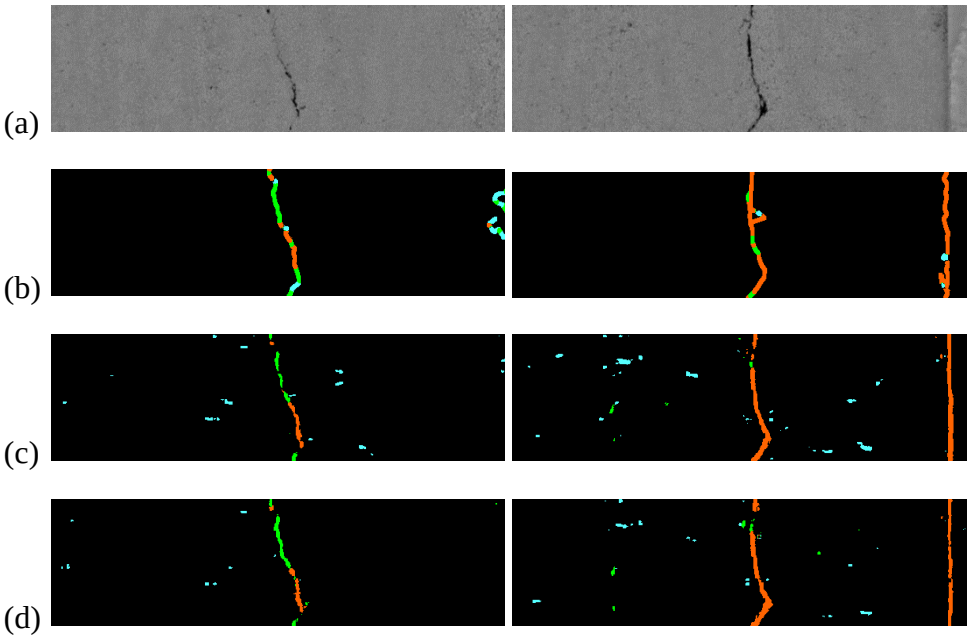


Figure 5.1.5 (a) Pavement Image, (b) Given Ground Truth from MTO, (c) Prediction from CNN with ADABOUND optimizer and weight (1, 1, 1), (d) Prediction from CNN with ADABOUND optimizer and weight (5, 3, 1) (Low: Cyan, Medium: Green, High: Orange)

5.2 The effect of optimizers on FCN

In the re-designed FCN, both the SGD optimizer and the ADAM optimizer was tested with a cross-entropy loss function. Although Long et al. (2016) showed a reasonable result with SGD optimizer, the ADAM optimizer was explored in this study to observe the effect of optimizers. In terms of the training time, there wasn't a remarkable difference between SGD optimizer and ADAM optimizer when the same number of epochs were used to train the

network. The sample prediction results from FCN with the SGD optimizer and ADAM optimizer is shown in Figure 5.2.1. As shown in Figure 5.2.1, FCN with either optimizer successfully captured all the crack segments that can be seen on the pavement image. Upon visually inspecting the predictions from these two optimizers, there wasn't any noticeable difference.

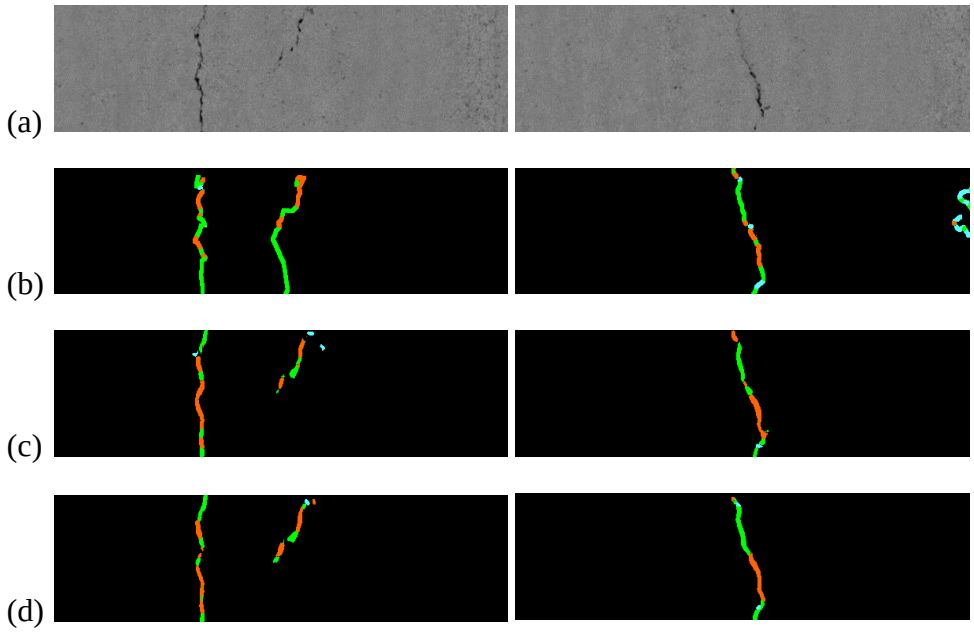


Figure 5.2.1 (a) Pavement Image, (b) Given Ground Truth from MTO, (c) Prediction from FCN with SGD optimizer (d) Prediction from FCN with ADAM optimizer (Low: Cyan, Medium: Green, High: Orange)

The numerical results from FCN with SGD optimizer and ADAM optimizer is shown in Table 5.2.1. Although the use of the ADAM optimizer showed a slight deterioration of precision in medium and high-level severity

compared with the SGD optimizer, it showed much better precision in low-level severity. ADAM optimizer achieved ~20% higher precision in low-level severity than the SGD optimizer. Additionally, recall in low-level severity improved by ~40%. Consequently, F1-score in low-level severity also improved by ~24%. However, there were no huge improvements observed in medium or high-level severity. Regardless of the different optimizers, the precision in medium-level severity remained around ~60%, and the precision in high-level severity remained around ~85%. This indicated that the effect of optimizers was minimal when the cracks were easily distinguished from the image. Conclusively, FCN performed better when it was trained with ADAM optimizer.

Table 5.2.1 FCN with SGD optimizer and ADAM optimizer

	Precision (%)			Recall (%)			F1-Score (%)		
	Low	Med	High	Low	Med	High	Low	Med	High
SGD	7.72	60.34	86.03	9.08	74.94	96.44	8.11	65.02	90.17
$ADAM$	27.50	59.59	84.23	49.08	89.53	98.63	32.08	68.82	89.89

5.3 The effect of optimizers and loss functions on U-Net

The U-Net was re-designed from scratch, and the selection of optimizers and loss functions were more flexible than FCN. Since the ADAM optimizer performed better than the SGD optimizer in FCN, the SGD optimizer was not tested with U-Net. Hence, U-Net was trained with the following combinations:

1) ADAM optimizer and cross-entropy loss function
2) ADAM optimizer and focal Tversky loss function
3) ADABOUND optimizer and focal Tversky loss function.

The numerical results of these U-Net configurations are shown in Table 5.3.1. As shown in Table 5.3.1, the biggest change in precision was shown from the cracks with low-level severities. There were minor changes in cracks with medium severity and cracks with high severity. Furthermore, the focal Tversky loss function performed better than the cross-entropy loss function when U-Net was trained with ADAM optimizer. Therefore, the focal Tversky loss function was selected to train U-Net with ADABOUND optimizer.

Changing the cross-entropy loss function to the focal Tversky loss function improved the precision of low-level severity by ~14%, but there wasn't a remarkable change in medium and high-level severities. Furthermore, when U-Net was trained with ADABOUND optimizer and focal Tversky loss function, there were minor improvements on all levels of severities. Conclusively, altering the optimizer and loss function on U-Net did have an effect on the cracks with low and medium severity, but the cracks with high-level severity stayed little above 80% in precision, regardless of different optimizers and loss functions.

Table 5.3.1 U-Net trained with ADAM optimizer and cross-entropy loss function; ADAM optimizer and focal Tversky loss function; ADABOUND optimizer and focal Tversky loss function

		Precision (%)			Recall (%)			F1-Score (%)		
Optimizer	Loss	Low	Med	High	Low	Med	High	Low	Med	High
ADAM	Cross-Entropy	18.76	54.70	84.53	29.28	79.27	98.40	20.76	61.61	90.08
ADAM	Focal-Tversky	32.61	57.68	82.04	52.46	83.95	98.48	37.21	65.65	88.45
ADABOUND	Focal-Tversky	34.12	61.31	82.60	61.42	87.62	98.70	40.06	69.97	89.07

In Figure 5.3.1, a couple of examples were taken from the test images. As shown in the left column of Figure 5.3.1, there were few noise pixels found

when the U-Net was trained with ADAM optimizer, but there weren't any noise pixels found when trained with ADABOUND optimizer. In general, U-Net predictions had a smoother shape of the cracks than the ground truth itself. From the test image shown on the left column of Figure 5.3.1, there were some dark noisy pixels found on the left side of the image, but U-Net successfully neglected these noises. Similarly, the test image shown on the right column of Figure 5.3.1 had noises on the right side of the image and U-Net did not classify any of these noises as crack pixels. Again, the localized shape of the crack segment was more naturally detected than the ground truth. Furthermore, the ground truth often had multiple alternating severity levels along the crack segment, especially when the crack had a curved shape. This behavior indicates that the ground truth was very sensitive to change in severity levels. Ultimately, U-Net prediction did classify the majority of the crack pixels into proper severity levels, but the sensitive ground truth restrained the precision to be higher. Conclusively, U-Net showed the best prediction results when it was trained with ADABOUND optimizer and focal Tversky loss function.

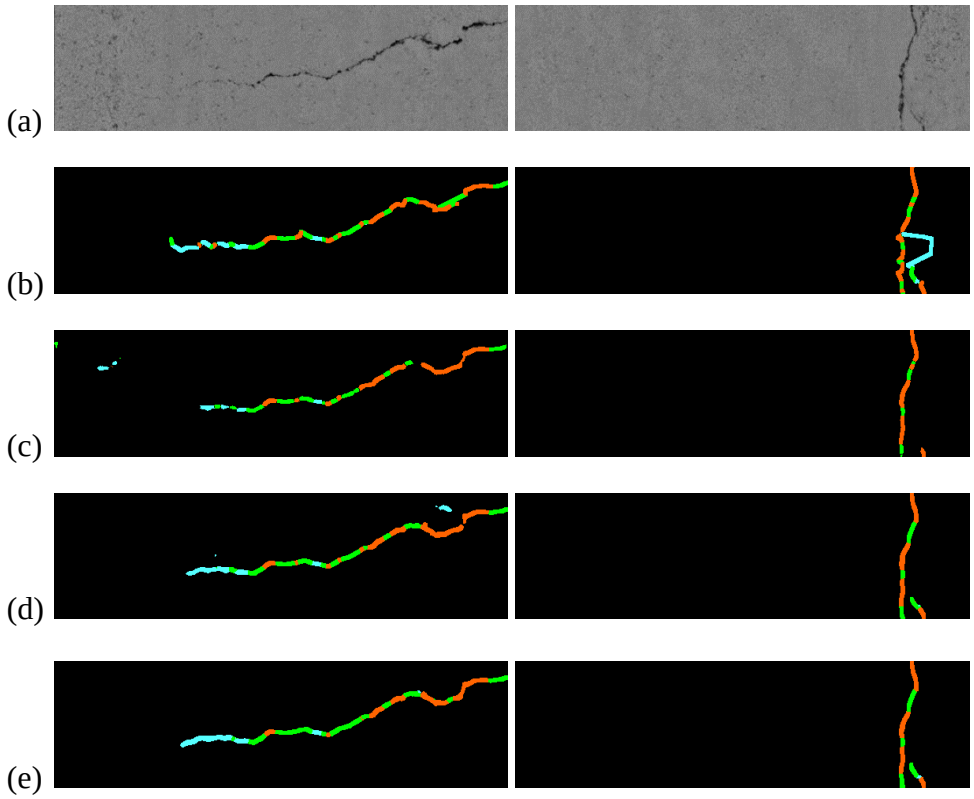


Figure 5.3.1 (a) Pavement Image, (b) Given Ground Truth from MTO, (c) Prediction from U-Net with ADAM optimizer and Cross-Entropy loss, (d) Prediction from U-Net with ADAM optimizer and Focal-Tversky loss (e) Prediction from U-Net with ADABOUND optimizer and Focal-Tversky loss (Low: Cyan, Medium: Green, High: Orange)

5.4 The effect of activation functions and optimizers on Attention U-Net

As proven from the experiments performed on U-Net, focal Tversky was proven to perform better than a typical cross-entropy loss function. Hence,

only the optimizers were changed in attention U-Net experiments to observe the effect of optimizers. Initially, the ADAM optimizer was used to train the attention U-Net and it produced a similar result as U-Net when it was trained with ADABOUND and focal Tversky loss function. When U-Net and attention U-Net was trained with identical optimizer and loss function, there was a slight improvement shown in attention U-Net, but the improvement was not as dramatic as expected. As observed in Table 5.4.1, different activation functions were attempted. Commonly, segmentation approaches including FCN and U-Net used softmax function because the last layer was a convolution layer with a 1x1 filter size, and each pixel in the image was convoluted once at a time. However, the use of the sigmoid function also served its purpose, but it did not show any noticeable changes from using a softmax function. Hence, for the sake of consistency with other segmentation approaches, softmax was preferred over the sigmoid activation function.

Table 5.4.1 Attention U-Net trained with ADAM optimizer and softmax function; ADABOUND optimizer and softmax function; ADABOUND optimizer and sigmoid function

		Precision (%)			Recall (%)			F1-Score (%)		
Activation	Optimizer	Low	Med	High	Low	Med	High	Low	Med	High
Softmax	ADAM	34.98	62.32	83.48	77.25	92.77	98.68	43.95	72.31	89.63
Softmax	ADABOUND	36.03	60.39	84.58	54.89	94.11	98.14	40.53	71.27	89.95
Sigmoid	ADABOUND	38.55	61.79	82.03	64.57	89.04	98.94	44.07	70.52	88.84

As mentioned earlier, the ground truth did not have precise labels for all the crack segments. As shown in Figure 5.4.1, there were discontinuous crack labels where it should have been continuous and vice versa. From the test image shown on the left column of Figure 5.4.1, the bottom left crack segment from the ground truth had labeled a part of it as high severity, but upon inspecting the test image, it looked more like a low or medium severity crack segment. Since the ground truth was generated using the raw data file that was collected based on multiple sensors including laser measurement, other aspects such as the depth of the cracks may have been deep enough to be classified as a high severity crack segment. However, the neural networks were only provided with a single channel camera simulated images, thus their resources were restricted to the pixel intensity. When the attention U-Net was

trained with ADAM optimizer and softmax function, the predictions were a little coarser than when the network was trained with ADABOUND optimizer.

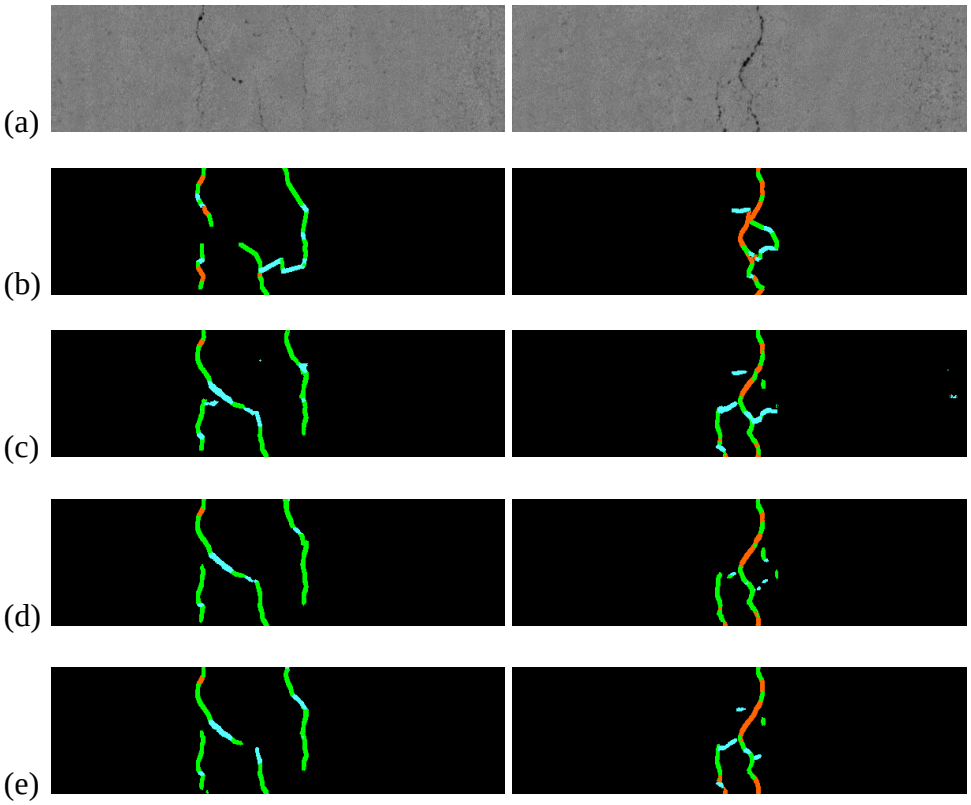


Figure 5.4.1 (a) Pavement Image, (b) Given Ground Truth from MTO, (c) Prediction from Attention U-Net with ADAM optimizer and softmax function, (d) Prediction from Attention U-Net with ADABOUND optimizer and softmax function (e) Prediction from Attention U-Net with ADABOUND optimizer and sigmoid function (Low: Cyan, Medium: Green, High: Orange)

The images shown in Figure 5.4.2 were extracted from Figure 5.4.1 to analyze the details of the prediction results. Although the precision itself did not have a

huge difference between different optimizers, some details differentiated between them. The prediction produced from the attention U-Net with ADAM optimizer had a spike outside the boundary of the crack segment, where the prediction produced from the attention U-Net with ADABOUND optimizer did not have such a spike. In general, the ADABOUND optimizer produced a more stable segmentation map than the ADAM optimizer, even though the numerical evaluation results were similar.

(a) (b) (c)

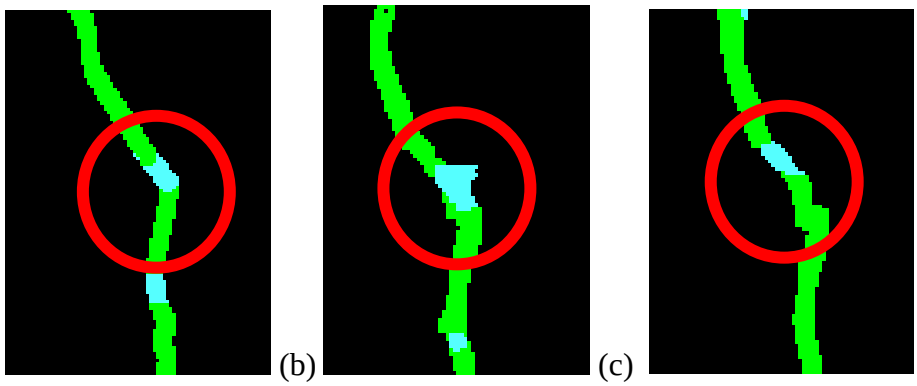


Figure 5.4.2 (a) Given Ground Truth from MTO, (b) Prediction from Attention U-Net with ADAM optimizer and softmax function, (c) Prediction from Attention U-Net with ADABOUND optimizer and softmax function (Low: Cyan, Medium: Green, High: Orange)

Chapter Six: Conclusion

In this research, various convolution-based deep neural networks were improved and re-constructed with various loss functions and optimizers to serve the purpose of crack detection from pavement images. To localize the cracks and classify their severity levels, the CNN with a structured prediction was improved by replacing the loss function with a weighted loss function to assign more weight towards the center pixel of the image patch. Moreover, the improved CNN with structured prediction performed multi-class classification, where the original approach was limited to the binary classification. With the weighted loss function, the network can effectively employ the spatial correlation between the pixels. After multiple experiments, the fixed weights of 5, 3, 1 showed a higher precision than the weights assigned by a bivariate normal distribution. The improved CNN with structured prediction showed the best performance when it was trained with ADABOUND optimizer and weighted cross-entropy loss function. Hence, the improved CNN with a structured prediction achieved the final precision of 12.02%, 42.28%, and 79.76% for cracks with the severity levels of the low, medium, and high, respectively. Consequently, it resulted in the F1 score of 19.63%, 55.60%, 85.89% for cracks with the severity levels of the low, medium, and high, respectively.

Although the improved version of CNN with a structured prediction served its purpose to fulfill the goal of this research, it had limitations of having false positive pixels around the actual crack segments and the average prediction time was as long as ~80 s to predict a single image of size 160x576 pixels because this approach required both pre-processing and post-processing. To resolve these issues, deep learning segmentation approaches including FCN, U-Net, and attention U-Net was re-designed and implemented for the first time in crack detection on paved roads. Since FCN, U-Net, and attention U-Net takes in the entire image and directly outputs a segmentation map of the same size as the inputted image, there wasn't any pre-processing or post-processing required for these approaches. Hence, on average, FCN took ~1 s, U-Net, and attention U-Net took ~0.1 s to predict an image of size 160x567 pixels. Also, there were far a smaller number of noise pixels detected through these approaches compared with the improved CNN with a structured prediction.

The FCN was implemented based on the pre-trained VGG-16, and as expected, FCN showed a smaller number of misclassifications and was free from noise pixels compared with the improved CNN with structured prediction. Upon analyzing the effect of optimizers on FCN, it showed better performance when it was trained with ADAM optimizer than the SGD

optimizer. Ultimately, FCN showed good precision of 27.50%, 59.59%, and 84.23% for cracks with the severity levels of the low, medium, and high, respectively. Additionally, it resulted in the F1 score of 32.08%, 68.82%, 89.89% for cracks with the severity levels of the low, medium, and high, respectively.

Furthermore, U-Net was re-designed and trained with different optimizers and loss functions. U-Net was trained with the following combinations: ADAM optimizer and cross-entropy loss function; ADAM optimizer and focal Tversky loss function; ADABOUND optimizer and focal Tversky loss function. Due to the nature of paved roads, the actual crack pixels only consisted of a few percentages of the entire image. Therefore, the focal Tversky loss function remediated the severe class imbalance issue. Conclusively, U-Net showed the best performance when it was trained with ADABOUND optimizer and focal Tversky loss function. Not only U-Net achieved a high precision of 34.12%, 61.31%, and 82.60% for cracks with the severity levels of the low, medium, and high, respectively, it also predicted the given test image much faster. Moreover, it resulted in the F1 score of 40.06%, 69.97%, 89.07% for cracks with the severity levels of the low, medium, and high, respectively.

Despite the good representational power from these networks, these architectures relied on multi-stage cascaded CNNs when the target varied in shape and size. These cascaded frameworks were used to extract a specific region of interest and make corresponding dense predictions on that particular region of interest. However, this cascaded process led to excessive and redundant use of computational resources and model parameters. For instance, multiple similar low-level features were repeatedly extracted by all models within the cascade. Thus, attention gates were introduced to the U-Net and this network was known as the attention U-Net. From the attention gates, the network can filter out the unnecessary region of the image and focus on the region of interest more efficiently. Along with the attention gates, the focal Tversky loss function was used to support solving the class imbalance issue. Hence, attention U-Net was tested with the following configurations: softmax function and ADAM optimizer; softmax function and ADABOUND optimizer; sigmoid function and ADABOUND optimizer. Among these configurations, attention U-Net showed the best performance when it was trained with softmax function and ADABOUND optimizer.

Although the attention U-Net achieved the highest precision amongst all the approaches attempted in this study, it did not show dramatic improvement from the U-Net results. The purpose of attention gates was to

filter out the unnecessary background and only focus on the region of interest, but unlike the original medical segmentation application of attention U-Net, the cracks are usually spread out through the entire pavement and the width of these cracks are usually very thin. Thus, these properties of cracks may have challenged the network to properly select the region of interest. Conclusively, attention U-Net achieved the highest precision of 36.03%, 60.39%, and 84.58% for cracks with the severity levels of the low, medium, and high, respectively. Also, it resulted in the F1 score of 40.53%, 71.27%, 89.95% for cracks with the severity levels of the low, medium, and high, respectively.

Although the dataset provided for this research was limited to the range of laser data, the results from different neural networks showed an outstanding performance. Therefore, the performance of the networks would potentially improve if they had RGB channel camera-based images to extract more valuable and realistic features. As previously mentioned, there were some uncertainties on the ground truth labels because they were generated based on the traditional machine learning algorithms. These uncertainties can be dealt with by manually inspecting each image and minimize the misclassifications. Alternatively, one can use the active learning algorithm (Settles, 2009) to improve the quality of the labels more efficiently. Essentially, the active learning algorithm excludes the uncertain labels and the

model is trained with the labels with high confidence. Hence, the labels with low confidence are set aside for human annotators to manually inspect them. Then, these inspected ground truth data is fed back to the model to improve the quality of the labels.

Furthermore, the potential for deep learning techniques in pavement application has been proven from this research. Hence, alternative neural networks can be applied for better performance. For instance, using multiple CNN models could potentially work better than a single CNN model. As an example, the concatenated multi-model CNN approaches such as 2D FCN (Cai et al., 2017), Holistically Nested 2D FCN Stage (Roth et al., 2018), and multi-model 2D FCN (Zhou et al., 2017) showed a competitive outcome in CT pancreas segmentation application. Thus, there are various deep learning architectures with different configurations that can be applied and there is no right answer for which network would work best for any specific applications.

Bibliography

(1) Abraham, N., and Khan N. M. "A novel focal tversky loss function with improved attention u-net for lesion segmentation." *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE, 2019.

(2) Amhaz, R., et al. "Automatic crack detection on two-dimensional pavement images: An algorithm based on minimal path selection." *IEEE Transactions on Intelligent Transportation Systems* 17.10 (2016): 2718-2729.

(3) Bahdanau, D., Cho, K., and Bengio, Y. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473* (2014).

(4) Bennett, C., Hernán, S., and Alondra, C. “Data collection technologies for road management.”, 2006.

(5) Cafiso, S., Alessandro, G., and Sebastiano, B. “Evaluation of pavement surface distress using digital image collection and analysis.”, *Seventh International congress on advances in civil Engineering*. 2006.

(6) Cafiso, S. et al. "From manual to automatic pavement distress detection and classification." *2017 5th IEEE International Conference on Models*

and Technologies for Intelligent Transportation Systems (MT-ITS). IEEE, 2017.

(7) Cai, J., Lu, L., Xie, Y., Xing, F., Yang, L. “Improving deep pancreas segmentation in CT and MRI images via recurrent neural contextual learning and direct loss function.”, In: MICCAI

(8) Chandra, A. L. “Perceptron: The Artificial Neuron.” *Medium*, Towards Data Science, 11 Aug. 2018, towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d.

(9) Davis, J. "Preventing and repairing potholes and pavement cracks." *Asphalt* 26.2 (2011).

(10) Fan, Z., et al. "Automatic pavement crack detection based on structured prediction with the convolutional neural network." *arXiv preprint arXiv:1802.02208* (2018).

(11) Gkovedarou, M., and I. Brilakis. "ROAD ASSET CLASSIFICATION SYSTEM." *2019 European Conference on Computing in Construction* (2019). DOI: 10.35490/EC3.2019.135

(12) Hubel, D., and T., W. "Shape and arrangement of columns in cat's striate cortex." *The Journal of physiology* 165.3 (1963): 559-568.

(13) Jing, L., and Zang A. "Pavement crack distress detection based on image analysis." *2010 International Conference on Machine Vision and Human-machine Interface*. IEEE, 2010.

(14) Jung, W., et al. “Exploitation of Deep Learning in the Automatic Detection of Cracks on Paved Roads.”, *Geomatica*, vol. 73, no. 2, 5 Oct. 2019, pp. 29–44., doi:10.1139/geomat-2019-0008.

(15) Kapela, R., et al. "Asphalt surfaced pavement cracks detection based on histograms of oriented gradients." *2015 22nd International Conference Mixed Design of Integrated Circuits & Systems (MIXDES)*. IEEE, 2015.

(16) Kingma, D. P., and Ba, J. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

(17) Krizhevsky, A, Ilya S., and Geoffrey H. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

(18) LeCun, Y., and Yoshua B. "Convolutional networks for images, speech, and time series." *The handbook of brain theory and neural networks* 3361.10 (1995): 1995.

(19) LeCun, Y., et al. "Backpropagation applied to handwritten zip code recognition." *Neural computation* 1.4 (1989): 541-551.

(20) Lin, T., et al. "Focal loss for dense object detection." *Proceedings of the IEEE international conference on computer vision*. 2017.

(21) Luo, L., et al. "Adaptive gradient methods with dynamic bound of learning rate." *arXiv preprint arXiv:1902.09843* (2019).

(22) Luo, L., et al. "Learning personalized end-to-end goal-oriented dialog." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019.

(23) Luong, M., Pham, H., and Manning, C. "Effective approaches to attention-based neural machine translation." *arXiv preprint arXiv:1508.04025* (2015).

(24) Mao, X., et al. "Hierarchical CNN for traffic sign recognition." *2016 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2016.

(25) McGhee, K. “Automated pavement distress collection techniques.”, Vol. 334. Transportation Research Board, 2004.

(26) Miller, J., and William Y. “Distress identification manual for the long-term pavement performance program.” No. FHWA-HRT-13-092, United States. Federal Highway Administration. Office of Infrastructure Research and Development, 2014.

(27) MTO, “Manual for Condition Rating of Flexible Pavements”, Distress Manifestations, http://www.ontla.on.ca/library/repository/mon/30005/334942.pdf (2016)

(28) Murakami, T., et al. “High spatial resolution LIDAR for detection of cracks on tunnel surfaces.” *CLEO: Applications and Technology*. Optical Society of America, 2018.

(29) Nair, V., and Geoffrey H. "Rectified linear units improve restricted boltzmann machines." *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010.

(30) Oktay, O., et al. "Attention u-net: Learning where to look for the pancreas." *arXiv preprint arXiv:1804.03999* (2018).

(31) Oliveira, H., and Paulo C. "Automatic road crack detection and characterization." *IEEE Transactions on Intelligent Transportation Systems* 14.1 (2012): 155-168.

(32) Ouyang, A., Chagen L., and Chao Z. "Surface distresses detection of pavement based on digital image processing." *International Conference on Computer and Computing Technologies in Agriculture*. Springer, Berlin, Heidelberg, 2010.

(33) Reddi, S. J., Kale, S., and Kumar, S. "On the convergence of adam and beyond." *arXiv preprint arXiv:1904.09237* (2019).

(34) Robbins, H., and Monro, S. "A stochastic approximation method." *The annals of mathematical statistics* (1951): 400-407.

(35) Ronneberger, O., Fischer, P., and Brox, T. "U-Net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and computer-assisted intervention.* Springer, Cham, 2015.

(36) Rosenblatt, F. "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* 65.6 (1958): 386.

(37) Roth, H.R. et al. “Spatial aggregation of holistically-nested convolutional neural networks for automated pancreas localization and segmentation.”, Medical Image Analysis 45, 94 – 107

(38) Rumelhart, D., Geoffrey H., and Ronald W. "Learning representations by back-propagating errors." *nature* 323.6088 (1986): 533-536.

(39) Settles, Burr. *Active learning literature survey*. University of Wisconsin-Madison Department of Computer Sciences, 2009.

(40) Shi, Y., et al. "Automatic road crack detection using random structured forests." *IEEE Transactions on Intelligent Transportation Systems* 17.12 (2016): 3434-3445.

(41) Simonyan, K., and Zisserman, A. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

(42) Sudre, C., et al. "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations." *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, Cham, 2017. 240-248.

(43) Swan, D. J., P. Eng, and M. Eng. "Fugro Roadware." (2014).

(44) Szegedy, C., et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

(45) Teomete, E., et al. “Digital image processing for pavement distress analyses.”, *Proceedings of the Mid-Continent Transportation Research Symposium*. 2005.

(46) Wilson, A. C., et al. "The marginal value of adaptive gradient methods in machine learning." *Advances in Neural Information Processing Systems*. 2017.

(47) Wu, Y., and He, K. "Group normalization." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

(48) Yu, S., et al. “3D reconstruction of road surfaces using an integrated multi-sensory approach.”, *Optics and lasers in engineering* 45.7 (2007): 808-818.

(49) Yu, Y., et al. “3D crack skeleton extraction from mobile LiDAR point clouds.”, *2014 IEEE Geoscience and Remote Sensing Symposium*. IEEE, 2014.

(50) Yusof, N. A. M., et al. "Crack Detection and Classification in Asphalt Pavement Images using Deep Convolution Neural Network." *2018 8th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*. IEEE, 2018.

(51) Zeiler, M., and Rob F. "Visualizing and understanding convolutional networks." *European conference on computer vision*. Springer, Cham, 2014.

(52) Zhang, W., et al. “Automatic crack detection and classification method for subway tunnel safety monitoring.”, *Sensors* 14.10 (2014): 19307-19328.

(53) Zhou, Yuyin, et al. "A fixed-point model for pancreas segmentation in abdominal CT scans." *International conference on medical image computing and computer-assisted intervention*. Springer, Cham, 2017.