EVALUATION OF DEEP LEARNING-BASED SEMANTIC SEGMENTATION APPROACHES FOR AUTONOMOUS CORROSION DETECTION ON METALLIC SURFACES

A Thesis

Submitted to the Faculty

of

Purdue University

by

Cheng Qian

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Civil Engineering

December 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF APPROVAL

Mohammad Reza Jahanshahi, Chair

Lyles School of Civil Engineering

Ayhan Irfanoglu

Lyles School of Civil Engineering

Shirley J Dyke

Lyles School of Civil Engineerings

Approved by:

Dr. Dulcy Abraham

Head of the School Graduate Program

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Qian,Cheng M.S, Purdue University, December 2019. Evaluation of Deep Learning-Based Semantic Segmentation Approaches for Autonomous Corrosion Detection on Metallic Surfaces. Major Professor: Mohammad R Jahanshahi.

The structural defects can lead to serious safety issues and the corrosponding economic losses. In 2013, it was estimated that 2.5 trillion US dollars were spent on corrosion around the world, which was 3.4% of the global Gross Domestic Product (GDP) [1]. Periodical inspection of corrosion and maintenance of steel structures are essential to minimize these losses. Current corrosion inspection guidelines require inspectors to visually assess every critical member within arm's reach. This process is time-consuming, subjective and labor-intensive, and therefore is done only once every two years.

A promising solution is to use a robotic system, such as an Unmanned Aerial Vehicle (UAV), with computer vision techniques to assess corrosion on metallic surfaces. Several studies have been conducted in this area, but the shortcoming is that they cannot quantify the corroded region reliably: some studies only classify whether corrosion exists in the image or not; some only draw a box around corroded region; and some need human-engineered features to identify corrosion. This study aims to address this problem by using deep learning-based semantic segmentation to let the computer capture useful features and find the bounding of corroded regions accurately.

In this study, the performance of four state-of-the-art deep learning techniques for semantic segmentation was investigated for corrosion assessment taskincluding U-Net, DeepLab, PSPNet, and RefineNet. Six hundred high-resolution images of corroded regions were used to train and test the networks. Ten sets of experiments were performed on each architecture for cross-validation. Since the images were large, two

approaches were used to analyze images: 1) subdividing images, 2) down-sampling images. A parametric analysis on these two prepossessing methods was also considered.

Prediction results were evaluated based on intersection over union (IoU), recall and precision scores. Statistical analysis using box chart and Wilcoxon singled ranked test showed that subdivided image dataset gave a better result, while resized images required less time for prediction. Performance of PSPNet outperformed the other three architectures on the subdivided dataset. DeepLab showed the best performance on the resized dataset. It was found Refinenet was not appropriate for corrosion detection task. U-Net was found to be ideal for real-time processing of image while RefineNet did not perform well for corrosion assessment.

1. INTRODUCTION

1.1 Motivation

Metals are widely used in civil infrastructure systems since they can offer more tensile capacity than other materials such as concrete and wood. However, the capacity of metals can decrease due to corrosion which can eventually lead to the collapse of the structures. Rehabilitation and replacement of a structurally deficient member represent not only a great expenditure to the owner, but also a significant indirect cost to the users due to partial or complete closure of the structure. A fast and efficient autonomous inspection system is needed to assess corrosion rate correctly. Therefore, the steel structures can be inspected more frequently, and more cost-effective maintenance can be performed to extend service lives of these structures.

1.1.1 Problem Statement

Steel is one of the most commonly used materials in civil infrastructure systems. American Institute of Steel Construction (AISC) claimed that steel, as a framing material for buildings, takes a 46% market share for non-residential and multi-story residential construction in 2017 [2]. The National Bridge Inventory (NBI) reported that more than 34% of highway bridges in the US use steel as structural supports [3].

However, these steel structures are not at good conditions. In 2016, 9.1% of the bridges in the US, or a total of 56,007, were suffering structural defects and needed to be repaired [4]. Corrosion as the primary cause of structural defects on steel [5] is causing a great amount of economical loss around the world. In 2016, National Association of Corrosion Engineers (NACE) conducted a study on the measurement of protection, application, and economy of corrosion internationally [1]. Based on the

measure, 2.5 trillion US dollars were spent on corrosion around the world, which was 3.4% of the global Gross Domestic Product (GDP) in 2013.

One main reason that have caused such great loss is that structures are not frequently inspected for corrosion assessment. For example, most highway bridges are typically inspected once every two years. Great amount of corrosion can develop during this long period. Section loss caused by corrosion on the critical members can greatly reduce the capacity of the bridge. Once the load rating, calculated by dividing capacity of the member by maximum live load on the member, is less than 1, actions will need to be taken [6]. The owner need to decide to either post a weight restriction, or to rehabilitate the bridge. Common rehabilitation practices will be adding steel plates or welding. If the capacity of the bridge is less than 3 tons, the critical member or even the whole bridge will be replaced [6]. These practices are very expensive, and require partial or complete closure of the bridge.

If the structures can be inspected more frequently, corrosion can be detected at an early the stage that does not affect the capacity of the structure. The low-cost maintenance can be performed at this stage: the rust will be cleaned and spot repainting will be applied to prevent further corrosion. A good coating on metal surfaces can protect the structure for up to 25 years [7], and the spot coating can elongate the service life of coating for around 10 years [8]. Based on life-cycle cost analysis (LCCA), the price of maintenance activity, such as washing and re-coating steel girders, is much smaller than the cost of one-time rehabilitation replacement [8]. Therefore, frequent inspection of steel structures can not only increase the life of the structures but also decrease the economic loss.

1.1.2 Objectives of the Research

Although more inspections are required to reduce the loss, current manual inspection procedures are time-consuming, labor-intensive, costly and in some cases dangerous. Inspection guidelines, such as National Bridge Inspection Standard, re-

quire inspectors to visually assess every critical member within arm's reach (i.e., two feet) [9]. However, in many cases, it is difficult to conduct a visual inspection of the structures, especially when they are at a high elevation or at water level. Inspectors need to hang from ropes or stand on a scaffold positioned on a boat to inspect inaccessible regions.

An alternative approach is to use robots to help engineers to inspect structures using high-resolution images. Bertino and Jahanshahi [10] conculded the framwork and challenges for collecting and qualifying the image data for civil infrastructures. An inexpensive approach is to use UAVs to inspect the structures [11, 12]. A $200 UAV can fly for 25 minutes and capture 1080p videos and images. Currently, there are several companies that provide engineers with service to capture high-quality images or videos. The images captured using UAVs can also be used for three-dimensional (3D) point reconstruction, and the defects can be assessed on the 3D models of civil infrastructures [13]. Reconfigurable swarm robots (RSR) can also be used for inspection at a low cost. Jahanshahi et al. [14] concluded the challenges and achievements of using RSR to monitor civil infrastructures.

Although advanced technologies enable engineers to collect data to visualize the whole structures, the more important issue is how to extract useful information from the captured images. It is still time-consuming and labor-intensive to have a human to assess corrosion in the images. The purpose of this study is to use state-of-art computer vision algorithms to assess visible corrosion on the metal surfaces.

1.2 Related Work

Several studies have used computer vision technologies to assess corrosion in images. Jahanshahi et al. [15], Ahuja et al. [16] and Spencer et al. [17] reviewed monitoring applications in civil engineering which have benefited from computer vision technologies and UAVs in recent years. These applications are not only used for damage detection but also for other inspection applications (e.g. structure component

recognition), and monitoring applications (e.g. measurement of static displacement and dynamic responses). To this end, there are mainly three kinds of techniques used for corrosion assessment: image classification, object detection, and semantic segmentation.

Image classification algorithms can be used to identify whether or not there is corrosion in an image. Livens et al. [18] and Pidaparti et al. [19] extracted features from the wavelet decomposition of corrosion images and then classified the images as pit formation or cracking using a Learning Vector Quantization (LVQ) network and fractal dimension (FD) analysis. Chun et al. [20] evaluated the degree of shadow cast by the flaky corrosion, and classified the corrosion in images into five grades, from early stage of rust to laminated flaky corrosion.

Object detection algorithms draw a box around the corrosion, therefore the location of the corrosion can be defined. Cha et al. [21] used a Faster Region-based Convolutional Neural Network (Faster R-CNN) to detect four types of defects on steel structures: corrosion at middle and high levels, and corrosion on bolt and steel delamination. Bounding boxes were drawn around the defect and labeled with the type of damage. Atha and Jahanshahi [22] subdivided images into small patches, and then used image classification to determine if there is corrosion in each image patch. Therefore, an approximate shape of the corrosion could be identified. However, since corrosion takes irregular shapes, image classification and object detection have difficulty finding the accurate area of corrosion.

The solution is to use semantic segmentation to draw an accurate boundary around the corroded region. Semantic segmentation algorithms have shown the ability to assess cracks from image and video data [23–28]. Since semantic segmentation algorithms identify accurate shapes of the corrosion, rate of corrosion can be evaluated by quantifying the corrosion at different rounds of inspection. Also, engineers can reinspect the surface to determine whether the corroded region has expanded since the last inspection [29]. Most corrosion assessments using semantic segmentation are

based on color or texture analysis since corrosion typically has a red-brown color and a rough texture.

The most commonly used color-based segmentation algorithm for corrosion segmentation task is thresholding. It means there will be a threshold value set to separate different kinds of pixels. Chen and Chang [30] used the neuro-fuzzy recognition approach (NFRA) to find the threshold values and then used image thresholding to segment corrosions from the grayscale images. Lee and Chang [31] performed statistical analysis on scatter plots of the RGB color space, and concluded that the most significant variables for rust defect recognition were the mean value of red, the difference value in green, and the difference value in blue. Shen et al. [32] used Fourier-transform to detect the defects on the coating of steel bridges, and then used K-Means Algorithm to recognize the rust defects on the coating.

Some studies also integrated the thresholding algorithm with other color-based algorithms to perform the segmentation task. Liao and Lee [33] detected corrosion on steel bridges by first detecting corrosion pixels using K-means algorithm on Hue (H) value of pixels, and then using double-center-double-radius (DCDR) algorithm on RGB and HSI color spaces to detect if there are more corrosion pixels in the rest of image. The detector could provide a good result on most images; however the accuracy was low when there was a large or dark rust area in the image. Diaz et al. [34] developed a Matlab program to detect corrosion in images. First, the red channel images extracted from RGB images of corrosion were transferred to grayscale images, then a threshold was set to segment corrosion from background. Although the approach reached a true positive rate of 90 percent, only 20 pictures were tested. More data are required to prove the accuracy and effectiveness of the program.

Texture analysis-based segmentation uses features such as roughness to detect corrosion. Xie [35] reviewed studies that use texture analysis methods to detect defects on metallic surfaces. Jahanshahi and Masri [36] evaluated the effect of different parameters, such as color space, color channel, and the window size of sub-images on

the wavelet-based texture analysis algorithms. It was shown that HSI color space was not suitable for the task, and CbCr color channels enhanced the performance.

Haralick et al. [37] used gray level co-occurrence matrices (GLCMs) to statistically analyze the frequency of different gray level combinations within a corrosion image. Some studies integrated GLCM analysis with color analysis to capture as much information as possible for corrosion detection [38], [39]. Bonnin-Pascual and Ortiz [38] developed a corrosion detection algorithm based on a cascade of weak classifiers. First, GLCM was calculated to measure the roughness of each patch of the image. The image patch with energy value higher than a threshold would be further inspected. Subsequently, pixels were classified based on Hue-Saturation-Value (HSV), which is a representation of Red Green Blue (RGB) color space. Medeiros et al. [39] calculated GLCM probabilities to measure the roughness, and analyzed Hue Saturation Intensity (HSI) color histogram to identify color change caused by corrosion. Then, the texture and color information were combined to optimize the performance of the classifier.

Another type of texture-based segmentation uses filter-based approaches. Ghanta et al. [40] performed a one-level Haar wavelet transform on the input images and then used entropy and energy values in the wavelet domain to train a Least Mean Squares classifier to determine if each 8×8 image patch is corrosion or not. Jahanshahi et al. [15] evaluated vision-based approaches for corrosion detection, it was concluded that both texture and color analyses should be applied to enhance the corrosion detection performance.

Some studies use machine learning algorithms to optimize the parameters for corrosion segmentation algorithms based on color and tecture analyses. Son et al. [41] first converted the images in the RGB color space to the HSI color space, then used machine learning algorithms, such as support vector machine (SVM) and back-propagation neural network (BPNN)to classify the pixels as corrosion or not. Shen et al. [42] trained an artificial neural network to allocate the corroded segments, which resulted in relatively high accuracy. But it was computational epensive, and it took

over fifteen seconds to process each image. Some studies, such as Livens et al. [18] and Ghanta et al. [40] used color or texture analysis to capture the features first, then used machine learning algorithms to segment pixels based on these features. Although the parameters in the algorithms are optimized through machine learning, these approcaes still require human engineers to define the features to be learned, which is subjective

In recent years, Convolutional Neural Networks (CNNs) [43], which will be explained in section 2, significantly enhanced the capabilities of computers for classification tasks. Cha et al. [21] and Atha and Jahanshahi [22] used CNN-based object detection algorithms to locate the corrosion, but failed to compute the accurate area of corroded regions.

Several semantic segmentation algorithms were designed based on deep convolutional neural networks (DCNNs) in recent years. Some researches also used these algorithms to segment corrosion on civil facilities. Ty et al. [44] used U-Net to detect the corrosion in the penstocks using the photos taken by a Micro Aerial Vehicle (MAV). The accuracy was low since there was large amount of noise in the images caused by the dark environment and water left in the penstocks. Nash et al. [45] compared the performance of the same Fully Convolutional Network (FCN) trained using a large dataset including 250 poorly labeled images and a small dataset including 10 images labeled by subject matter experts. It was shown that the quantity of the images in the dataset is more important than the quality of the labels. Hoskere et al. [46] developed a multi-scale parallel DCNN architecture. The architecture was first used as a damage classifier to segment the defects on the structures into six different classes, including steel corrosion, and then used a damage segmenter to differentiate between pixels that represent damage and background. It was shown that damage segmenter successfully reduced the possibility of false detection. Although there has been a few studies where CNN algorithms are used to segment corrosion in images, most of them only applied the most basic semantic segmentation architectures, such

as DCNN and FCN, while there have been many CNN-based semantic segmentation networks developed for other applications.

1.3 Scope

The objective of this study is to investigate state-of-the-art deep learning techniques for semantic segmentation for the corrosion assessment task, so that better-informed decisions for maintenance of metal structures can be made. Section 2 introduces the principles of deep learning techniques used for semantic segmentation. Section 3 describes the four algorithms used and evaluated in this study. Section 4 describes the training process, including data generation and fine-tuning for the semantic segmentation model. Section 5 gives an explanation of the evaluation metrics and statistical analysis used in this study. Section 6 presents sample results obtained from the models and evaluates the performance of four models based on Intersection over Union (IoU), precision and recall scores, and cost of computation. Section 7 summarizes the main conclusions of this study and suggests improvements for further work.

2. PRINCIPLES OF DEEP LEARNING-BASED SEMANTIC SEGMENTATION

In this study, semantic segmentation is used to let the computer classify each pixel in the image as corrosion or background. In this chapter, the principles of deep learning-based semantic segmentation algorithms will be explained along with their use in corrosion recognition.

2.1 Fully Convolutional Network (FCN)

The most basic deep learning-based semantic segmentation algorithm is called FCN [47]. It was designed by UC Berkeley in 2014. It was trained on the dataset called PASCO VOC [48]. This dataset consists of 6,929 images for semantic segmentation, and twenty classes including humans, animals, vehicles, and indoor furniture. FCN showed an Intersection over Union (IoU) score, which evaluates the accuracy of semantic segmentation algorithms, of 62.2% accuracy on the test set of PASCO VOC.

The input to the network is an image, where the image is represented by a tensor with the size of W×H×3, where W and H are the original width and height of the input image, and 3 represents Red, Green and Blue (RGB) values. The output of the network is a W×H binary image. For corrosion assessment task, each white pixel shows that the corresponding pixel in the input image is corrosion, and the black pixel shows the corresponding pixel in the input image is background. The network includes multiple convolution, activation, pooling, and transpose convolution layers that are introduced below.

2.1.1 Convolution Layer

Convolution layers use kernels to slide over the input image to extract features and store them in the feature map. Figure 2.1 shows a 2-D example of convolution using a 2×2 kernel on a 4×4 input at the stride of 2 to produce a 4×4 feature map. The values in the kernel are first multiplied with the corresponding values in the yellow region of the input. And the sum of the products are recorded in the corresponding yellow region of the output feature map. For instance, the 1 in the output is calculated by $0\times1 + 1\times2 + 0\times3 + 0\times4 = 2$. Then, the kernel moves two steps to the right. The same calculation for the numbers in the green region give the output of 3. After the kernel slides over the whole input image, the output feature map is generated. In order to keep the output at the same size as input, zero padding is added to the input. For instance, as shown in Figure 2.2, zeros are first padded around the input. Then, a 3×3 kernel convolves over the 4×4 input at the stride of 1 to produce a 4×4 feature map. The same computation takes place between the values in the input and the kernel including zero padding.

Changing the values in the kernel, which are called weights, helps the network to capture different features during convolution. For instance, as shown in Figure 2.3, the first kernel outputs the embossed objects and helps to capture texture information. The second kernel outputs the outline of the object, enabling extraction of object boundaries. The third kernel outputs the blurred image, which decreases the difference between adjacent pixels. Kernels will be optimized during the training process, to help the network find the most useful information. The algorithm of optimizing the weights will be discusses later in this chapter.

The size of the kernel is usually very small compared to the input image (e.g., 3×3, 5×5, or 7×7). So, the first few convolution layers can only capture low-level features such as lines, curves, or edges. The next few convolution layers can combine these small features to learn parts of the boundary or the texture. These features help the network to recognize the whole object.

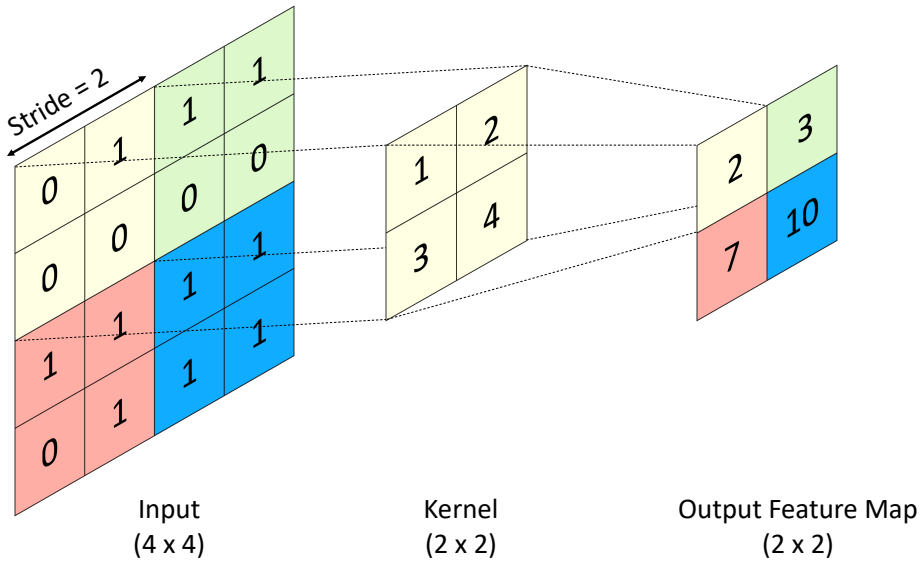


Figure 2.1.: 2-D example of convolution using a 2×2 kernel, on a 4×4 input at the stride of 2 to produce a 4×4 feature map.

2.1.2 Activation Function

The values, stored in the feature maps, representing the features captured by the kernels are called activations. Since not all the features extracted from the images are useful for the task of detection, activation functions are used after the convolutions to define which features are important to the task [49]. In the semantic segmentation tasks dealing with RGB images, negative activations are not useful. Therefore, an activation function that turns all the activations to positive values should be used. The activation function used in most of the CNN based architectures is ReLU [50]. It is used since the computation is very simple, which is suitable for the complex CNN networks. The expression of the ReLU function is:

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{2.1}$$

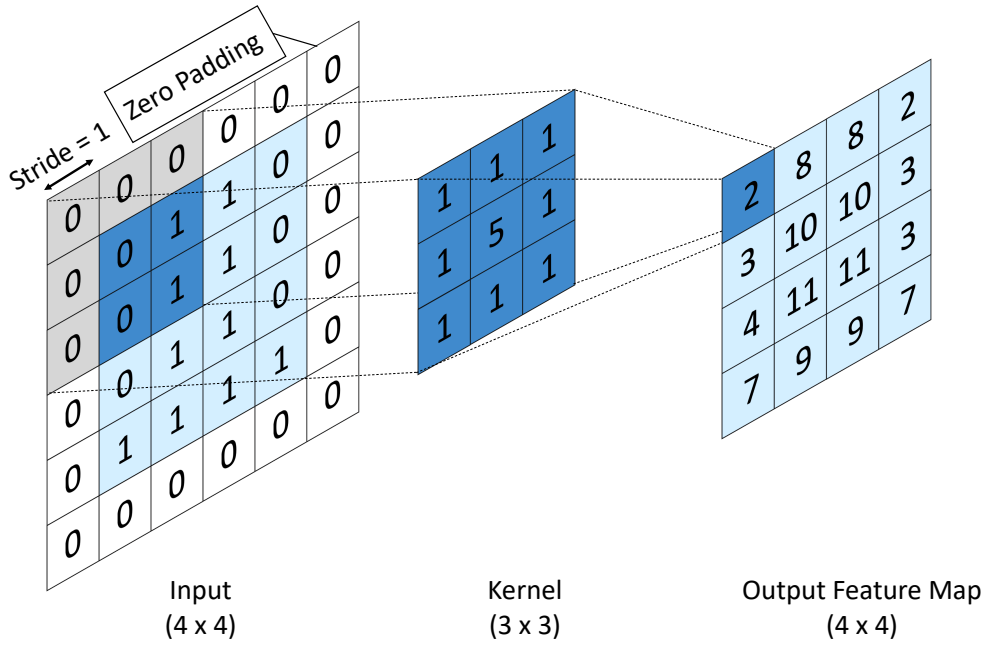


Figure 2.2.: 2-D example of convolution with zero padding: a 4×4 input is convolved using a 3×3 kernel at the stride of 1 to produce a 4×4 feature map.

It means that if the number in the output feature map tensor is less than zero, then it will be replaced by zero. However, if the number is greater than zero, the number could keep its value.

An FCN architecture can be built by adding up the convolution layers, as shown in Figure 2.4. D means D feature maps generated by D kernels at each step. Since there are only 2 classes, corrosion and background, in the corrosion detection task, the feature maps at the last step has the size of W×H×2. Activation functions can be added between convolution layers to improve the results. Since the output prediction should have the same size as the input image, the sizes of the feature maps are kept the same. And many convolution layers are added to make sure that enough features can be captured. The consequence of this is a large amount of computation.

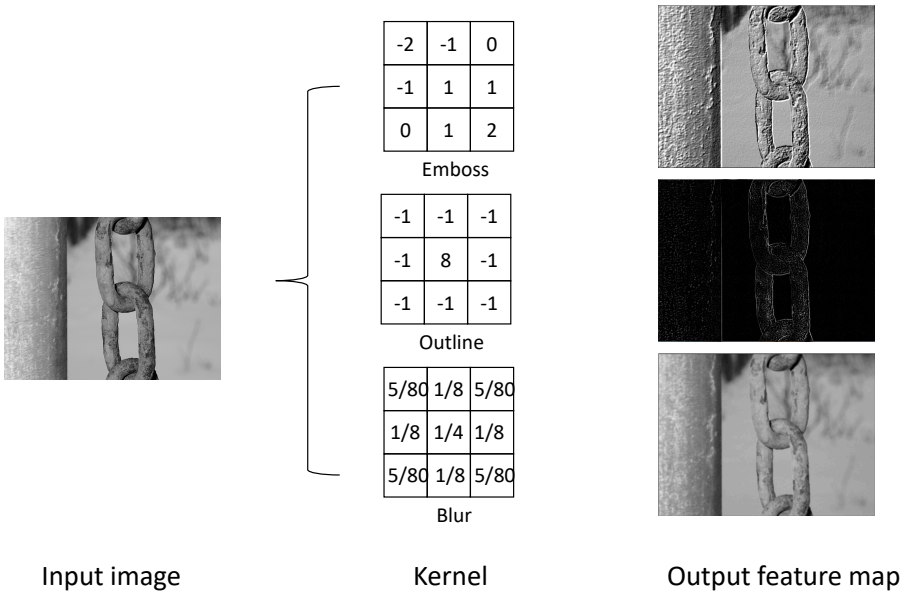


Figure 2.3.: Sample convolution using different kernels.

2.1.3 Down-Sampling and Up-Sampling

In order to reduce the amount of computation for FCN architectures, the feature maps are first down-sampled to a small size, at a low resolution, and then up-sampled to the original size at a high resolution, as shown in Figure 2.5.

Pooling

The down-sampling process is called pooling. There are two kinds of pooling: max pooling and average pooling. Figure 2.6 shows an example of max pooling with 2×2 kernel at stride of 2. As the figure shows, the greatest value in every 2×2 patch is stored in the corresponding region in the output feature map. For example, the greatest value in the upper left, yellow region, is 6, so, 6 is stored in the upper left corner of the output feature map. Average pooling is similar to max pooling. Instead of exporting the maximum number to output, the average of the values in each patch

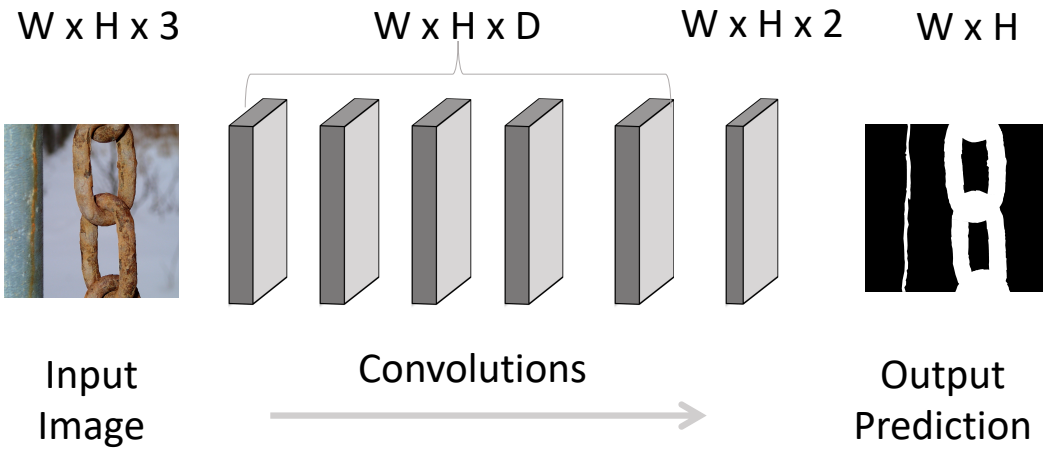


Figure 2.4.: FCN architecture composed of convolution layers, while keeping the size of the feature maps the same as the input image.

would be exported. Most semantic segmentation algorithms use max pooling since the most important features can be preserved. Average pooling is not preferred since it counts all the features including those not important for the detection task [51].

Transpose Convolution

There are several methods of up-sampling. Transpose convolution the most popular one since it can learn how the convolution layers can up-sample the feature map. Figure 2.7 shows an example of a 2-D transpose convolution where a 2×2 kernel at the stride of 2 is used to up-sample a 2×2 input feature map to the size of 4×4. The values in the kernel are first multiplied with the value in the upper left corner of the input where these products are recorded in the corresponding yellow region of the output feature map. Then, the kernel moves to the right, and the same computation

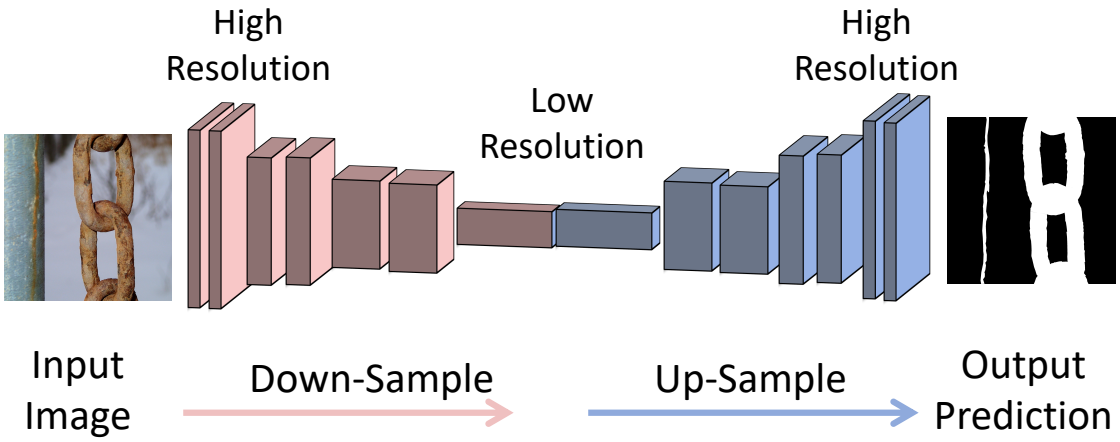


Figure 2.5.: Down-sampling and up-sampling: the feature map is down-sampled to a lower size then up-sampled back to the same size as the input image.

takes place for the values in the green region. Since the kernel convolves at the stride of 2, the values result from multiplication are stored in the green elements, which are two steps right to the yellow ones. After the kernel slides over the whole input image, the output feature map is generated. In some cases, the kernel convolves the feature map at the stride of 1, which means the new values are stored one step right to the previous ones. In this case, there will be some overlap when storing the values in the output. The sum of the product values will be taken at these regions.

2.2 Prediction

After building up the network to capture and decode the features, activation functions are used to find the probability of each pixel being corrosion. Because the activations are expected to be the probabilities in the range of zero to one, activation

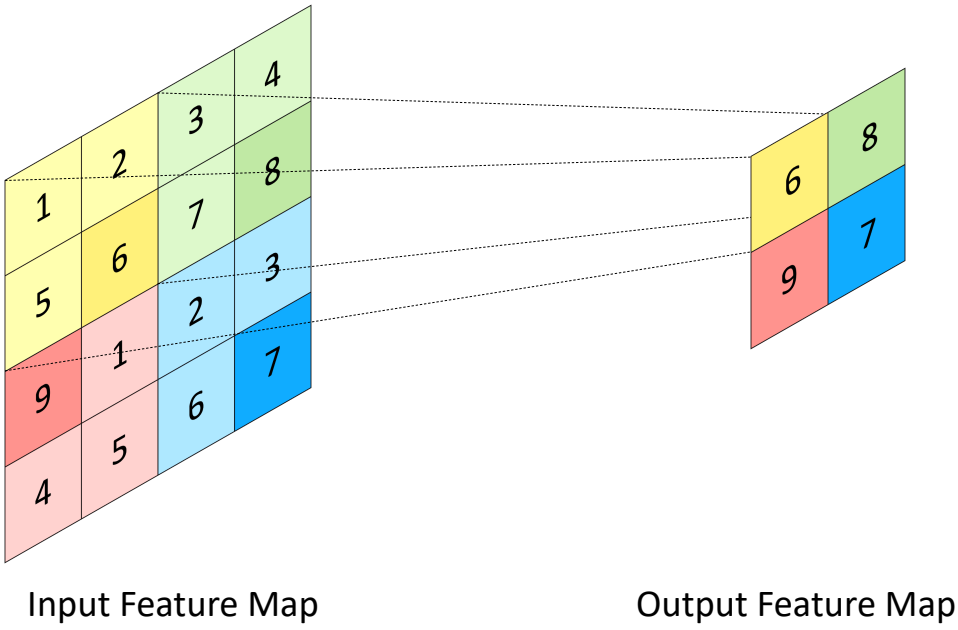


Figure 2.6.: Sample of max pooling process in 2-D. The max value in the input feature in a 2×2 patch is stored in the output feature map.

functions that can transfer the activations to the values between zero and one should be used. The two most commonly used activation functions to give outputs of the network are the sigmoid function and the softmax function that can be calculated by Equations 2.2 and 2.3. Sigmoid function is used for binary classification. S in the function represents the probability, and z represents the output value for the pixel calculated by the network. Softmax function is used for multi-class classification. S_j computes the probability for each class. z_j represents the output value for a given pixel identified as class j, and K represents total number of classes. Since the corrosion detection task is a binary classification task, K for this task is 2.

$$S_z = \frac{1}{1 + e^{-z}} \tag{2.2}$$

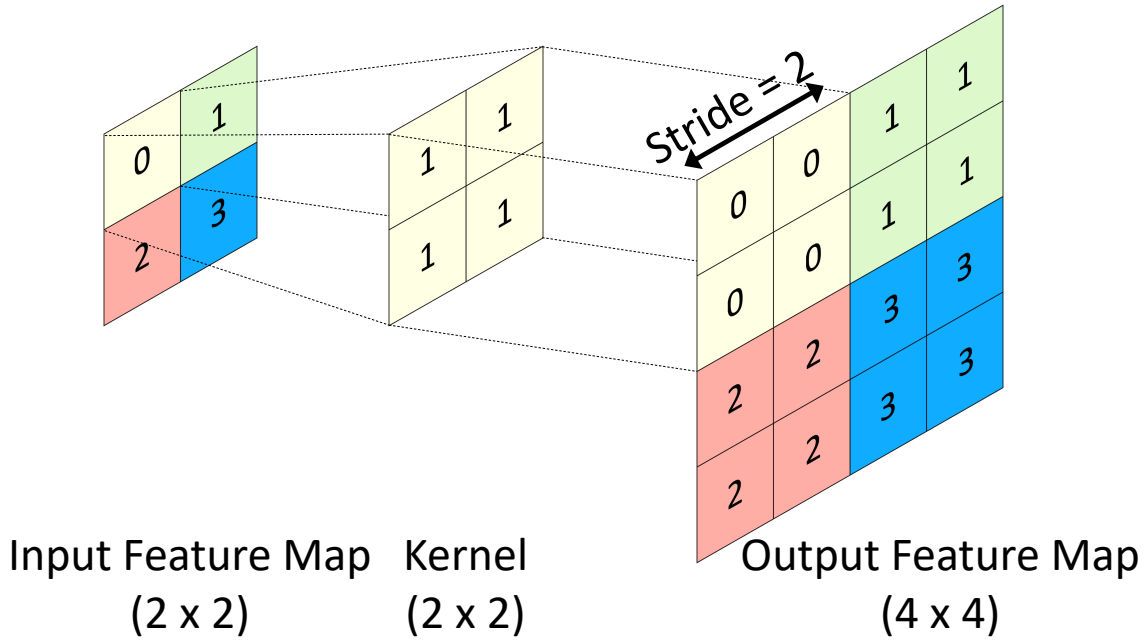


Figure 2.7.: 2-D example of transpose convolution using a 2×2 kernel, on a 2×2 input at the stride of 2 to produce a 4×4 feature map.

$$S_j = \frac{e^{z_j}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.3}$$

2.3 Training

After computing the probability of each pixel being corrosion, the probability is compared with the expected results. The network will be trained to minimize the difference. For example, for a corrosion pixel, the expected prediction is 1. However, the probability calculated from the network could be 0.6. The network will be trained to increase this probability.

2.3.1 Loss Function

Loss function is calculated in order to find how the network should be trained. It is a function that quantifies the difference between the prediction and the ground truth.

$$J = -y * log(S) \tag{2.4}$$

Cross-entropy loss [52] is the mostly used loss function for deep learning. It can be calculated using Equation 2.4, where y indicates ground truth label of the pixel, and S indicates predicted probability of the pixel being y. A large value of loss means the error is large, and the model performs poorly. In order to get better models, the weights in kernels should be trained to minimize the value of loss function J.

2.3.2 Gradient Descent

The process to minimize loss is called optimization. The optimization algorithm used for semantic segmentation is gradient descent [53]. It is a iterative process. Figure 2.8 shows a 1-D representation of gradient descent of loss function J controlled by the weight w. After passing the images through the network, the loss can be calculated, represented by the point on the loss function J. The loss is expected to decrease in the direction of the slope at the point at step α. The new loss can be calculated using Equation 2.5, where α represents a learning rate that controls the length that each step of gradient descent will have. In order to increase the speed of training, several images are set together as one batch to be processed. After training over one batch the weights of kernels will be updated. This process will be repeated until the loss value is minimized. Training over one batch is called one step, and training over all training data is called one epoch.

$$w_n := w_{n-1} - \alpha \frac{\mathrm{d}J(w)}{\mathrm{d}w} \tag{2.5}$$

The optimization process is much more complicated than the 1-D example shown in Figure 2.8. It can be very slow and inefficient to use a constant learning rate for

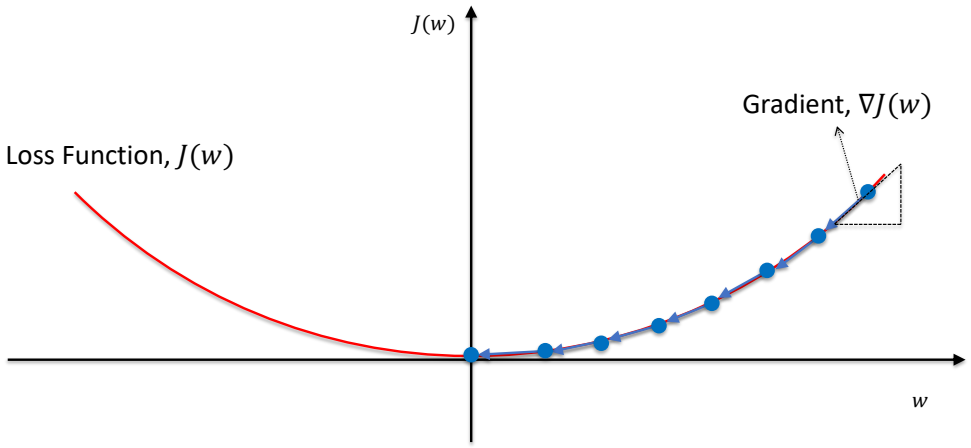


Figure 2.8.: Sample of gradient descent on a 1-D loss function J controlled by the weight w.

gradient descent. Therefore, several optimization functions built based on gradient descent are developed to update the learning rate after every iteration to increase the speed of training [54]. The optimizers used in this study are Momentum Optimizer [55], Adam Optimizer [56], and RMSProp Optimizer [57]. Momentum Optimizer adds a vector calculated from previous step to the current step to increase the speed of training. RMSProp optimizer uses the average of square root of past gradient descent. Adam optimizer includes average of exponentially decaying gradients while updating the gradients. The direction and the size of gradient descent depends on the previous step, and therefore the network converges at a more accurate direction.

2.3.3 Back Propagation

Back propagation is the process of adjusting the weights in the whole network based on gradient descent. Using gradient descent analysis, the direction and amount of expected changes of the activations in the last layer can be determined. However, the changes of activations are controlled by the weights in the kernels and the activations in the previous layer. Therefore, the expected change in activations is determined first. Then, the weights will be adjusted to make these desired changes, and the expected change in the last second layer is recorded. This calculation propagates backward from the last layer to the first layer of the network, which is the input image. As a result, the adjustments in all the weights will lead to the expected activations in the last layer determined by gradient descent.

2.4 Network Architectures

Performance of the network can be primarily affected by the arrangement of convolution, pooling and activation layers. There are many architectures of convolution neural networks (CNNs) built to allow the model to perform the pixel classification task better. These architectures can be used in down-sampling and up-sampling processes to help improve the performance of semantic segmentation.

Two architectures will be used in this paper to identify the corroded regions better. The first one is Visual Geometry Group-16 (VGG16) [58]. Visual Geometry Group at the University of Oxford developed this architecture to win the ImageNet Large Scale Visual Recognition Challenge in 2014 (ILSVRC) [59]. The model is composed of five convolution blocks: In the first two convolution blocks. There are two convolution layers with a kernel size of 3×3, and one max pooling layer with kernel size of 2×2, and stride of 2. The last three convolution blocks are composed of three convolution layers with a kernel size of 3×3, and the same max pooling layer as previous two blocks. After the input image passes through these five blocks, two fully connected layers are used to transfer the feature maps to the size of $1 \times 1 \times N$, where N is the

number of classes that need to be classified. At last, an activation function is used to predict the class of the image.

Another architecture that is used in this study is the Residual Networks (ResNet) [60]. Microsoft research team developed this architecture to win ILSVRC in 2015 (ILSVRC) [59]. In order to get better results, several architectures are designed to be very deep. However, these models are complicated to train since there are many parameters to be optimized. When optimizing the model, the gradient that back propagated through the model to update weights disappears before the first few convolution layers, which is called vanishing gradient. In ResNet, the residual block is designed to solve this problem and allow the network to be deep. The equation of the residual block is given in Equation 2.6:

$$y = F(x) + x \tag{2.6}$$

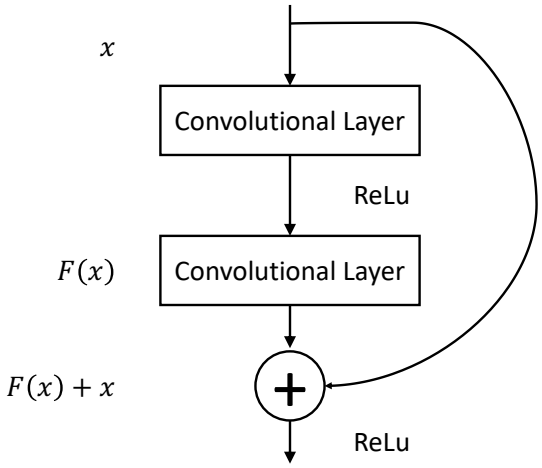


Figure 2.9.: Single residual block in ResNet.

Equation 2.6 means that the input feature map is added to the output of the feature map passed by convolution layers and activation functions. As shown in Figure 2.6, in each residual block, there is one convolution layer, followed by a Relu function, and another convolution layer. Based on the number of total convolution layers in the whole network, the network has the name ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet- 152. For example, ResNet-18 includes 18 convolution layers. Although these networks have different depths, all of them follow a similar stream. The feature map first has half of the input image size, then 1/4, 1/8, 1/16 of the input, and ends up with the size of 1/32 of the input image. At the end of the network, the average pooling layer is connected to the output of the last convolution layer. Then, the fully convolution layer and activation function are applied to get the predictions. As the number of layers increases, the number of parameters needed to be trained increases. The ResNet network should be chosen based on the size of the training data set.

3. SEMANTIC SEGMENTATION ALGORITHMS

Although FCNs can be used for semantic segmentation problems with the strategy of down-sampling and up-sampling to save the amount of computation, there are still some issues that prevent the FCNs from achieving a good performance. First, the models are very slow to be trained since there are several duplicated computations caused by the similarity among the neighboring pixels. Second, the location information of the pixels can be lost during the process of down-sampling and up-sampling. In order to resolve these issues, many models are designed based on FCNs.

Several deeep learning-based semantic segmentation algorithms have been designed for PASCO VOC [48] to detect humans, animals, vehicles, and furniture. Some algorithms are designed for automatic driving to identify different kinds of objects in the images of streets. Some are designed for bio-medical research to recognize cells or vessels. Several algorithms have shown good performances for these tasks, so this research aims to evaluate their performance on the corrosion assessment task.

Most of the algorithms are designed only for good detection performances, such as SegNet [61], Deep Lab series [62], and PSPNet [63]; some are designed for small amounts of training data, such as U-NET [64]; some are designed for high resolution images, such as RefineNet [65]; and some are designed for real-time segmentation, such as ShuffleNet [66], LinkNet [67], and MultiNet [68]. Four algorithms are selected for the segmentation of a varity of corroded region in high resolution images. These algorithms are discussed in this chapter.

3.1 U-Net

U-Net [64] is a simple but efficient model that won the International Symposium on Biomedical Imaging (ISBI) challenge in 2015. As the winner, U-Net showed the

ability to segment the cells in 512×512 pixel images in less than one second with the support of a graphics processing unit (GPU). In the competition, only 30 images were provided for training. U-Net achieved the IoU score of 0.9203 and 0.7756 for two different datasets provided in the challenge. It showed the segmentation ability where limited amount of training samples are available, and also where the cells in the images have different shapes.

U-Net is named after the shape of the model. The network structure is shown in Figure 3.1 and represents the down-sampling and then up-sampling process of the architecture. On the left hand side of the network, the feature map is down-sampled to the size of $\frac{1}{256}$ of the original image, where there is a concern that too much information is lost during this process. The feature maps in each block before pooling are concatenated to the up-sampled feature maps at the same level. This helps the model to preserve the information and better combine the location and RGB information of the pixels.

3.2 DeepLab V3+

DeepLab is a series of semantic segmentation algorithms, developed in recent years. DeepLab V3+ [69] is the most recent version. It achieved the IoU score of 85.7% while testing on the PASCAL VOC testing set. DeepLab was designed to solve two problems of FCN: First, the down-sampling process reduces the resolution of the feature map. Second, it is difficult for FCNs to detect both large and small objects.

In order to adress these two issues, two strategies were implemented in DeepLab. First, instead of a sequence of the process of down-sampling, convolution, and then up-sampling, atrous convolution was used. Figure 3.2 shows a 2-D example of atrous convolution using 3×3 kernel, at the atrous rate of 1 and stride of 1. The kernel skips one pixel during the convolution process. So, using traditional convolution, as shown in the left hand side of the figure, a 3×3 kernel can, for instance, extract features from the patches with the size of 3×3. However, using atrous convolution as shown

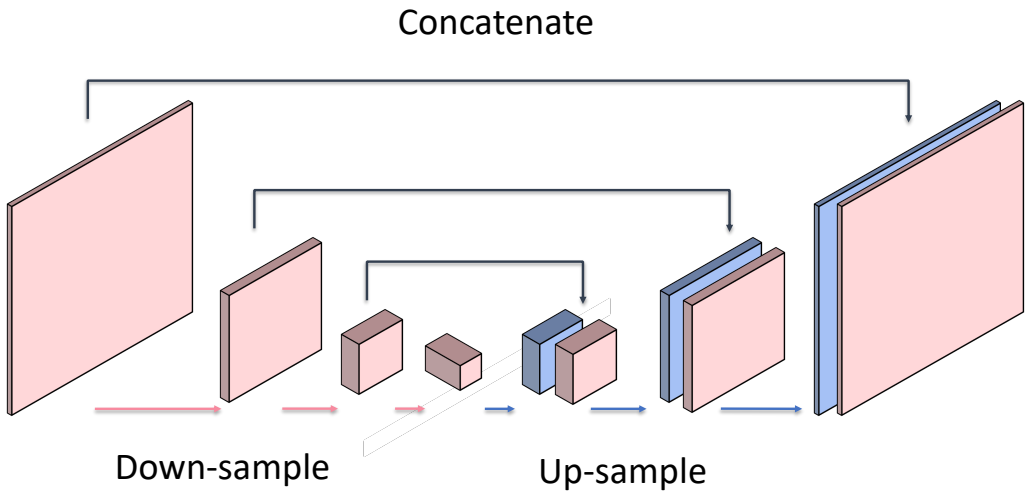


Figure 3.1.: U-NET architecture: the feature maps in each block before pooling are concatenated to the up-sampled feature maps at the same level.

in the right hand side of the figure, a 3×3 kernel can only extract features from the patches with the size of 5×5. With zero padding, the resulting feature map will have the same size as the input image. Compare to down-sampling and up-sampling process in FCNs, where only part (i.e., $\frac{1}{4}$) of information is preserved after pooing, atrous convolution extracts information from every pixel. Therefore, the resolution of the feature map resulting from atrous convolution will be higher than that of the feature map resulting from down-sampling and up-sampling process.

In order to solve the second problem, the strategy called atrous spatial pyramid pooling (ASPP) is used. As shown in Figure 3.3, input feature map is convolved with the kernel at the same size, 3×3, but at different rates of 6, 12, 18 and 24. The four different output feature maps are then merged together. This strategy allows the kernels to capture the features at different levels. Therefore, DeepLab

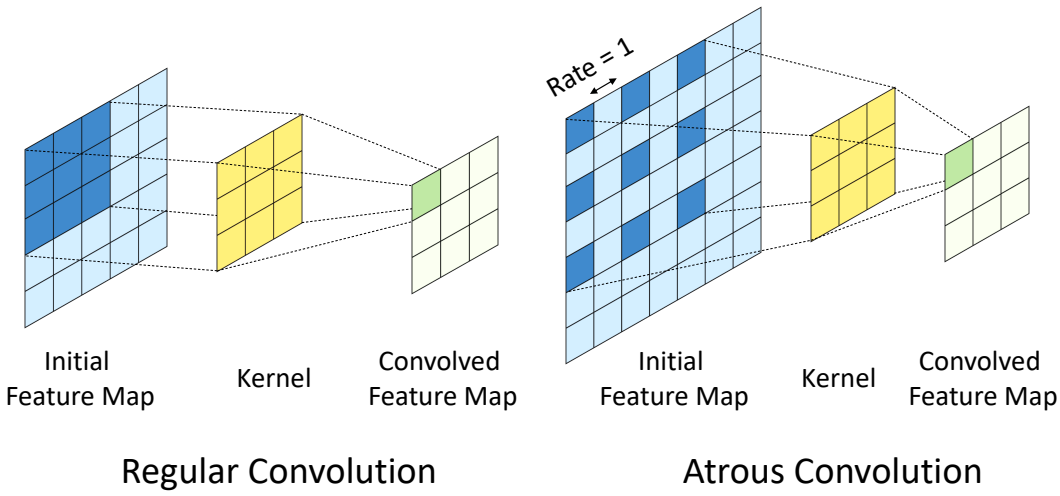


Figure 3.2.: Sample of Atrous Convolution in 2-D.

can capture the shapes of the objects with different sizes, while preserving the very detailed information.

3.3 PSPNet

A pyramid scene parsing network (PSPNet) [63] is another good model for semantic segmentation. Using this network, the IoU score reached 82.6% on the PASCAL VOC testing set. PSPNet is designed to solve three issues: first, the relationship between the objects and their neighbors or background is difficult to learn; second, there are some categories that easily confuse the identifier; third, shows an example of a 2-D transpose convolution where a 2×2 kernel at the stride of 2 is used to up-sample a 2×2 input feature map to the size of 4×4. PSPNet is chosen in this study since the third problem may occur in corrosion detection tasks: some corrosion parts are so small that the identifier may not be able to detect them. Also, there

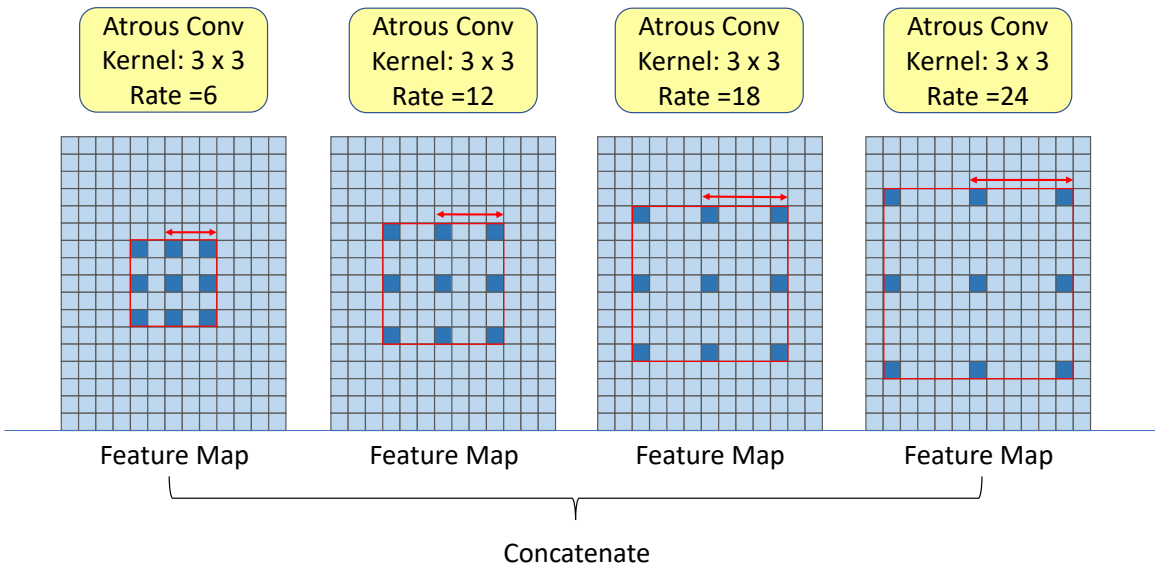


Figure 3.3.: 2-D representation of Atrous Spatial Pyramid Pooling, using 3×3 kernel, at the rate of 6, 12, 18 and 24.

are many cases in which a small uncorroded region is bounded by corrosion where the classifier may missclassify this region as corrosion. In order to solve these issues, PSPNet is designed to fuse the information of small areas with the information of the whole image so that the relationship between the objects and background can be learned.

PSPNet uses a pyramid pooling module as shown in Figure 3.4. Max pooling is performed in this module. In order to obtain the information with different receptive fields, the input feature map is pooled into different bins: 1×1, 2×2, 3×3, and 6×6, as the example shown in the figure. The feature map of size 1×1 is obtained from global average pooling of the whole image where the 1×1 bin preserves the information from the whole image. Each value in a 2×2, 3×3 and 6×6 feature map captures the information from $\frac{1}{4}$, $\frac{1}{9}$ and $\frac{1}{36}$ of the whole image. The output feature maps are up-sampled to the same size as the input feature maps and are concatenated

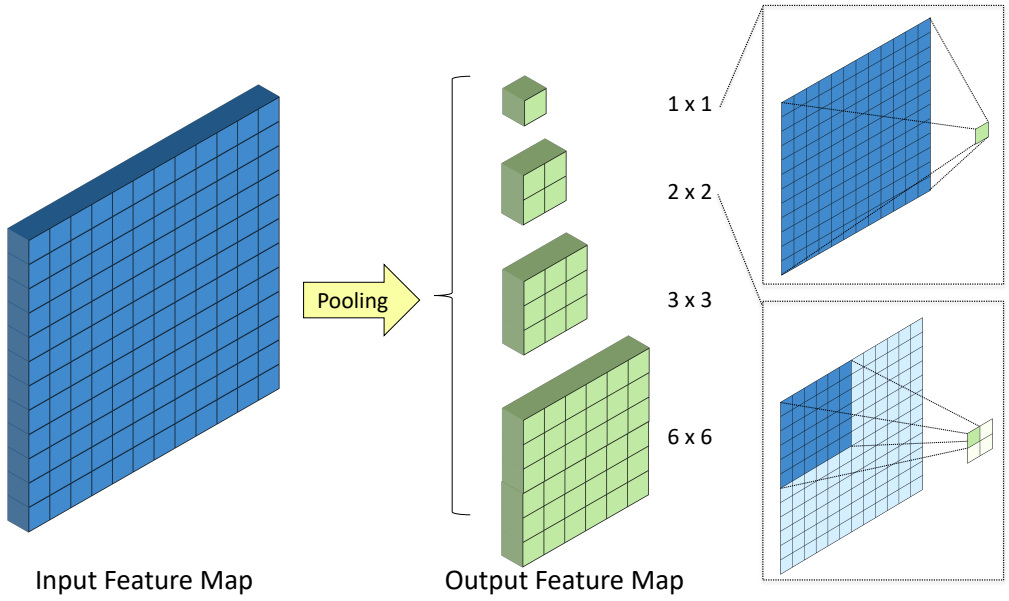


Figure 3.4.: Four-level pyramid pooling used in PSPNet

together with the input feature maps. The pyramid pooling module helps the network to capture the information from different regions of the image so that the relationship between the pixels is better preserved and both large objects and small objects can be detected accurately.

3.4 RefineNet

Although DeepLab used atrous convolution to solve the problem of losing resolution during the down-sampling process, the high-resolution feature maps require large GPU memories to support computations. In order to resolve this issue, a cascaded network is designed and the name of the model is called RefineNet [65]. The network achieved the IOU score of 83.4% on the PASCAL VOC testing set. RefineNet multi-resolution fusion block, as shown in Figure 3.5, was used to fuse the feature maps at different resolutions to help the network to gather features at different levels.

The left hand side of the figure shows feature maps at different down-sampling levels (i.e., 1/4, 1/8, 1/16, and 1/32 of the input image). The smallest feature map with the size of 1/32 of the input image is first convolved and then up-sampled to the size of 1/16 of the input image. Then, the resulting feature map is added to the feature map with the size of 1/16 of the input image. The resulting feature map is convolved, up-sampled, and added to the larger feature map until the resulting feature map has the size of 1/4 of the input image. This block helps the model to fuse the feature maps with different resolutions together in order to reduce the loss of resolution caused by down-sampling [65].

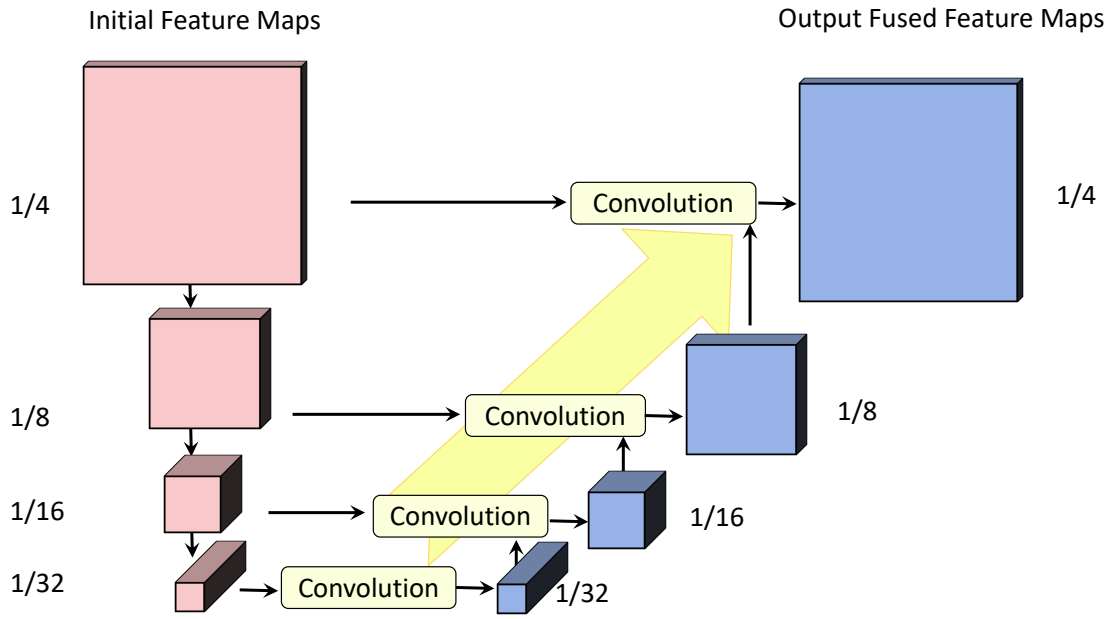


Figure 3.5.: Multi resolution fusion block in RefineNet architecture: Feature maps 1/4, 1/8, 1/16 and 1/32 representively fused together from lower level to high level

3.5 Transfer Learning

Although the four algorithms have improved the efficiency of computation from FCNs, they are still complex. There are over 10 millions parameters to be trained in each network. However, the information from the images in the dataset is limited. This can lead to a issue, overfitting [70]. It means the network fits the training data too well, and cannot make correct prediction on new data. It is likely to happen when a complex network is traind using a small dataset, because there are limited information to be learned.

Transfer learning [56] is an effective method to prevent overfitting. It means the model is already trained for another task on a large dataset. For example, in this study, a pre-trained model of ResNet is employed in PSPNet and refined as the encoder. The ResNet model was pre-trained on the ImageNet dataset [71]. The data-set includes 1.4 million labeled images that are categorized into 1000 classes. Since the original dataset is quite large and covers very board classes, it can already capture many generic features before training, so training based on the pre-trained model using a relatively small learning rate can enhance the training performance of a network. As a result, transfer learning uses a pre-trained model to help training a better model in a shorter time.

4. TRAINING PROCESS

4.1 Dataset

The dataset used in this study consists of 600 images. 399 images were captured on the Purdue campus using different digital cameras. The resolutions of these images were 3024 × 4032, 3024 × 3024 or 2592 × 1944 pixels. The rest of the images were downloaded from the Internet. These images have a variety of sizes, and most of them were larger than 2000 × 2000 pixels.

4.1.1 Data Generation

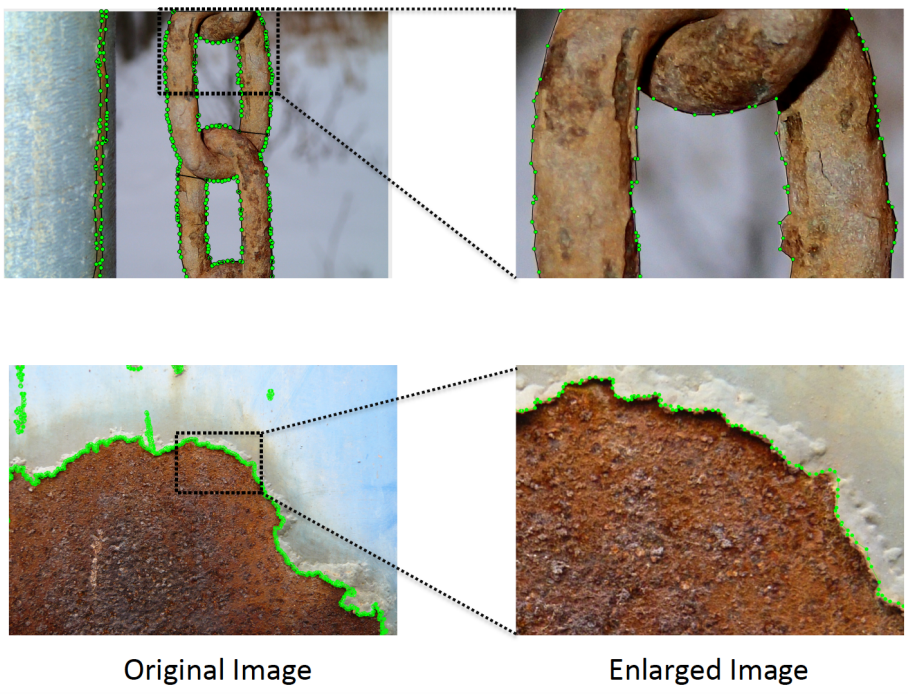


Figure 4.1.: Image polygonal annotations generated by connecting short lines along the boundary of corrosion.

The labels were made manually by connecting short lines along the boundary of corroded regions, as shown in Figure 4.1. The area bounded by the lines is corrosion, and the remaining parts of the images are defined as background. The images were zoomed in several time to make sure that each boundary is accurately drawn. For labeling, a pixel that represents corrosion is set to one; otherwise, it is set to zero.

4.1.2 Computing Platform

All the experiments were performed on a server with a Linux system. All the four networks were implemented in TensorFlow [72]. Tensorflow is a popular open source machine learning environment developed by Google. Since deep learning algorithms are computationally expensive, four state-of-art Nvidia Titan X Pascal GPUs were used to perform training and testing. GPUs were used in this study for calculation instead of Central Processing Unit (CPU), the nerve center of a computer, because it consumes less memory and is much faster in speed. Nvidia Titan X Pascal GPUs used for this study has a memory capacity of 12 GB, and can perform the calculation at the speed of 11.4 Gbps.

4.1.3 Preprocessing the Data

Since the images in the dataset had a pixel size greater than 2000×2000, a GPU with the memory size of 12GB could not process the whole image at a time. In order to perform the test on the available GPUs, the images were pre-processed to have the size of 512×512 or smaller. Two methods were chosen to process large images. First, the images were subdivided into 512×512 sub-images. For example, a 2048×2560 image, would be subdivided into twenty 512×512 training images. Since the original images had different sizes, some of the processed images had smaller sizes. After deleting some images with petite sizes (e.g., 12×512), the subdivided dataset included 12,854 images. Subdividing the large images enabled the dataset to be relatively large. Alternatively, the images were resized to the size of 512×512, so

no matter how large the original image was, the output image after resizing always had the size of 512×512. This method allows the model to view the whole image, so that the shape of the corrosion parts could be learned. Another advantage of this method is that the whole image could be segmented at one time, and there is no need to merge the small images in order to see the whole image. The disadvantage of this method is that information was lost during resizing. Since two different methods of pre-processing images have different advantages and disadvantages, both data-sets were used for training and testing and the results of the two datasets were compared.

4.1.4 Cross-validation

In order to perform statistical analysis on the models, cross-validation was employed so that variability of the models could be evaluated. The dataset was first divided into five equal subsets so that for the subdivided dataset, there would be 2,569 images in each subset; and for resized dataset, there would be 120 images in each subset. Then, for each test, four subsets were used as training data, and the remaining one subset was used as testing data. The subset used as testing data was shuffled so that the subdivided dataset and the resized dataset each had five sets of training and testing data.

Since five sets of data are not enough for the statistical evaluation of the performance of the models, another five sets of data were generated by randomly distributing 80 percent of images into the training set and the rest into the testing sets. As a result, ten sets of data were used for training and testing of each model, for both subdivided and reseized datasets.

4.1.5 Image Augmentation

Since the number of images in the resized dataset was small, the training set was augmented to enlarge the dataset to avoid overfitting. The following augmentations were applied to the training set: rotation within the range of 40 degrees; width

and height shift with the range of 20 percent; shear within the range of 20 percent; horizontal flip; and zoom within the range of 20 percent. Data augmentation enlarged the training set to include 0.5 million images.

4.2 Training Parameters

All the training parameters were set to the same values as those suggested in the original papers. The values of learning rate and number of epochs were changed based on the value of loss during the training. The previous studies used a base learning rate of train the model to learn at a fast speed. However, the the optimizer can overshoot the minimum loss and fail to find the optimal set of weights. Therefore, a smaller learning rate can be used to find the global minimum loss and the trade-off is the long training time [73]. To find the most suitable training rate for the experiment, the loss was recorded during the training process. When the loss of the training decreases at first few epochs of training but remained relatively constant during the rest of training, the learning rate would be set as one-tenth of the original value. Then, the same training process would be conducted, and the IoU score of the validation data would be calculated. If the IoU score increased, then the lower learning rate would be used for training. Training parameters used for different models are summarized in Table 4.1. The batch size was set based on the GPU memory capacity that can process these images at one time. A suitable optimizer could help the model approach the best hypothesis efficiently. Adam, momentum, and RMSProp, as described in Chapter 2, are the three most commonly used optimizers for deep learning tasks. Since these optimizers all perform well for the task, the choice of the optimizer was based on the original papers. However the base learning rate used for the optimizer was fine-tuned based on training loss.

Table 4.1.: Parameters used during the training process for the four models

	U-NET	**DeepLab**	**PSPNet**	**Refinenet**
Training Steps	10000	20000	30000	8400
Training Batch Size	5	12	4	5
Optimizer	Adam	Momentum	Momentum	RMSProp
Loss Function	Sigmoid Cross Entropy	Softmax Cross Entropy	Softmax Cross Entropy	Softmax Cross Entropy
Base Learning Rate	1 e^{-5}	1 e^{-4}	1 e^{-4}	1 e^{-7}

5. STATISTICAL EVALUATION

5.1 Evaluation Metrics

For semantic segmentation, accuracy is not a fair metric to be used to evaluate different algorithms. Instead, intersection over union (IoU), recall, and precision are chosen to evaluate the four semantic segmentation algorithms discussed in Chapter 4. IoU is the most popular evaluation metric for semantic segmentation algorithms.

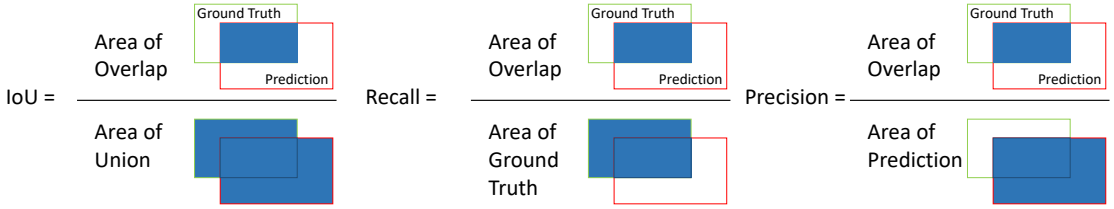


Figure 5.2.: Graphical representation of evaluation metrics.

As shown in Figure 5.2, IoU is calculated by dividing the area of overlap by the area of union of the ground truth and prediction. A poor prediction, which means there is little overlap between ground truth and prediction, results in a low IoU. A good prediction, which means the prediction is very similar to the ground truth, results in a high IoU.

The other two evaluation metrics used are recall and precision. These two metrics help users to define the best architectures for different applications. Recall is the area of overlap between ground truth and prediction divided by the area of ground truth. A low recall value means several corroded regions are not recognized by the network. Therefore, if the user wants to make sure as much corrosion as possible is detected, the architecture resulting in a high recall value should be used. Precision is the overlap area divided by prediction. A high precision score shows that relatively

small number of background pixels are mistakenly detected as corrosion. Therefore, precision scores should be compared if the user wants to get a high quality prediction.

In order to calculate the areas for corrosion and background with irregular shapes, a confusion matrix need to be introduced. Confusion matrix is a table that is typically used while evaluating machine learning algorithms. A binary confusion matrix used in this study is shown in Figure 5.3. If the the result of prediction is corrosion, then the prediction is said to be positive(P). Likewise, if the the result of prediction is background, then the prediction is said to be negative(N). If the prediction is correct, then the result is said to be true(T); if the prediction is incorrect, then the result is said to be false(F). Therefore, there will be four kinds of results. True Positive (TP) means the corroded pixel is classified as corrosion; False Positive (FP) means the uncorroded pixel is detected as corrosion, which is a wrong prediction; False Negative (FN) means the corroded pixel is not detected as corrosion by the model, which is also a wrong decision; True Negative (TN) means the background is correctly detected as uncorroded pixel.

After dividing the results into four different classes, the evaluation metrics can be calculated by Equations 5.1, 5.2 and 5.3. For instance, the area of overlap between ground truth and area of union can be calculated by counting the number of TP pixels in the image. And area of union can be calculated by counting the number of TP, FP, and FN pixels in the image. Then IoU can be calculated. The equations of these functions are given by:

$$IoU = \frac{TP}{TP + FP + FN} \tag{5.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

		Ground Truth: Corrosion	Ground Truth: No Corrosion
Prediction	Corrosion	TP	FN
Prediction	No Corrosion	FP	TN

Figure 5.3.: Confusion matrix used for corrosion assessment, where TP represents True Positive, FP represents False Positive, FN represents False Negative, and TN represents True Negative.

5.2 Box Plots

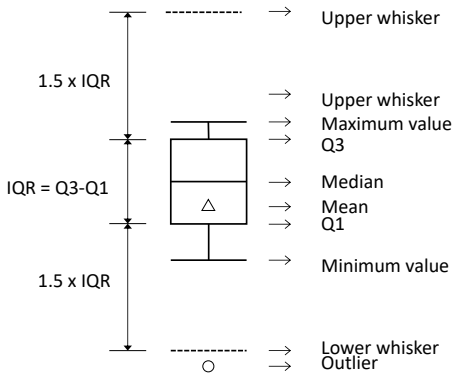


Figure 5.4.: Sample of Box Plot.

Box plots [74] are used to help visualize the results and to help make statistical decisions. IOU, precision, and recall scores are plotted in Figure 5.4. Each box in the figure shows the value of ten sets of scores calculated based on ten sets of data, evaluating the performance of each model trained and tested using subdivided or

resized datasets. The line in the box shows the median of the scores, and the triangle in each box represents mean. Upper bound and lower bound of each box are the third quartile (Q3) and the first quartile (Q1) values, respectively. Since Q1 represents the median between the minimum value and the median of the scores, and Q3 represents the median between the maximum value and the median, the variance of the scores can be observed from the size of the box. If the interquartile range (IQR), calculated by (Q3 - Q1), is large, then the variance of the scores is significant. The lines above and below the box show the maximum and minimum scores, respectively. The circles represent the outliers that are greater or smaller than the upper or lower whiskers, where the upper whisker is calculated by Q3 + 1.5IQR, and the lower whisker is calculated by Q1 - 1.5IQR.

5.3 Test for Significant Difference

The inferential statistic is used to find the best architecture by testing whether there is a difference between two sets of results.

5.3.1 Wilcoxon Signed Rank Test

Wilcoxon signed-rank test [75] is used in this study to test whether there is a significant difference between two sets of data, when the small samples do not have a normal distribution. For the Wilcoxon test, the differnce between pairs of data in the samples are calculated. If there are N data x in each sample, the difference between pairs of data are calculated and ranked from small to large so that each pair has a rank R_i . Then, statistic W can be calculated using Equation 5.4. Table 5.1 shows an example of Wilcoxon Signed Rank Test on two samples x_1 and x_2. Assuming x_1 and x_2 are two sets of sample results to be compared. In order to calculate statistic W, first, difference between each sample result, $(x_{1,i} - x_{2,i})$, is calculated. Then, the absolute values of the differences are ranked from 1, recorded as R_i. Lastly, W

is calculated by summing up the products between $(x_{1,i} - x_{2,i})$ and R_i as shown in Equation 5.4. If W is large, then there is a great difference between two samples.

Table 5.1.: Example of Wilcoxon Signed Rank Test on two samples x_1 and x_2

i	$\mathbf{x}_{1,i}$	$\mathbf{x}_{2,i}$	$(\mathbf{x}_{1,i} - x_{2,i})$	$\mathbf{R}_i$
1	10	10	0	1
2	11	10	-1	3
3	12	10	-2	6
4	13	15	2	6
5	14	15	1	3
6	15	15	0	1

The p-value for this test is the probability of getting statistic W if a new sample is randomly generated and compared with x_1. It can be either determined using Wilcoxon single ranked table [76], or using the built-in functions in the mathematics or statistics software (i.e., Python) The similar samples will result in a high p-value. In order to make statistical decision, the p-value is compared with a critical p-value. The critical p-value used for most researches is 0.05. Thus, when the p-value is lower than 0.05, there is a significant difference between two groups of data.

$$W = \sum_{i=1}^{N_r} [sign(x_{2,i} - x_{1,i}) \cdot R_i] \tag{5.4}$$

6. EXPERIMENTAL RESULTS AND DISCUSSION

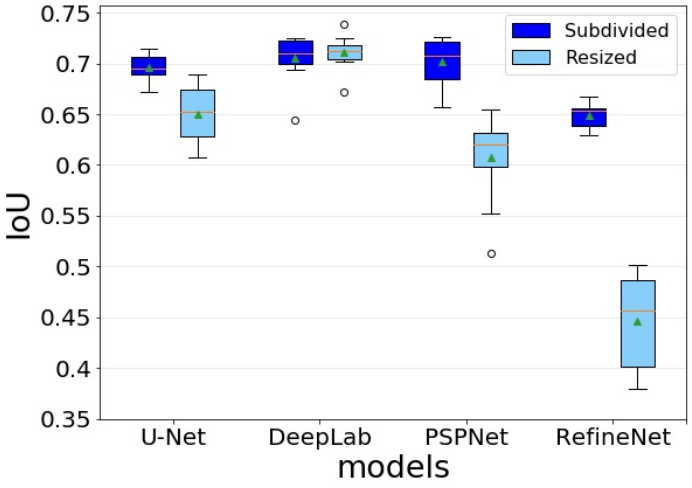
In this chapter, the performances of four state-of-the-art semantic segmentation models trained on subdivided and resized datasets are evaluated based on IoU, precision, and recall scores. Also, the time required to segment corroded region, GPU memory required to perform the test, and number of parameters in each network are compared.

6.1 Performance Evaluation of Semantic Segmentation Models

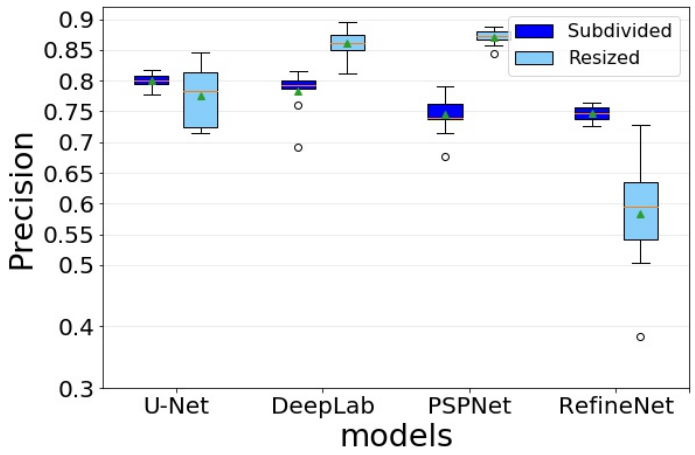
Performance of different networks can be visualized and compared using the box charts in Figure 6.1. The dark blue boxes are the results for the subdivided dataset, and the light blue boxes are the results for the resized dataset.

6.1.1 IoU Scores

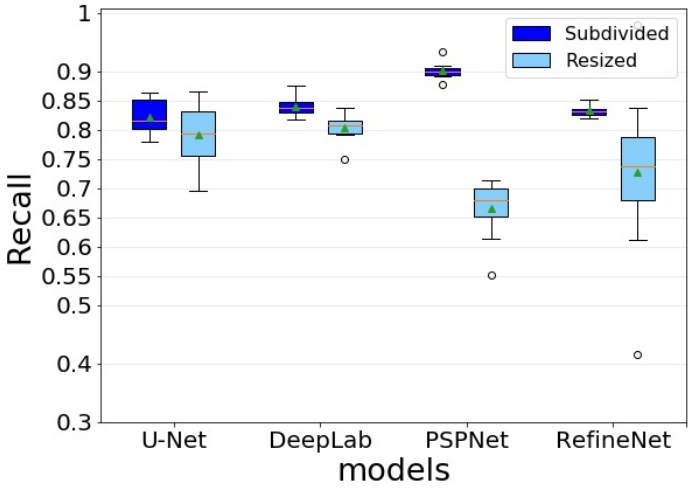
The first chart shows the results of IoU scores. It is obvious that RefineNet performs poorly on the resized dataset where the mean value of IoU scores is less than 45%, and the interquartile range of the results was around 8%, which is huge. So RefineNet is not recommended to be used on the resized dataset. The conclusion is reasonable since RefineNet is originally designed for high-resolution images. Too much information is lost during the resizing process. Therefore, RefineNet failed to extract useful information from the resized images. For U-Net and PSPNet, the mean IoU scores for resized dataset are lower than the subdivided dataset. The Wilcoxon test confirmed this observation. Both sets had a p-value much smaller than 0.05, which means there is a significant difference between the results for the two sets. Thus, for U-Net and PSPNet, the subdivided dataset is preferred. It is interesting to note that DeepLab results in a little bit higher mean IoU score, and less variance on the resized



(a)



(b)



(c)

Figure 6.1.: Performance of the four state-of-art semantic segmentation models trained on subdivided and resized images: (a) IOU scores, (b) precision scores, and (c) recall scores.

dataset than on the subdivided dataset. However, the Wilcoxon test showed that there is no significant difference between the sets. Therefore, users can choose to use the subdivided or resizsed dataset based on different applications when using Deeplab Network. Since for most networks the subdivided dataset leads to better results, the comparison of different architectures will be made for the subdivided dataset, plotted in dark blue boxes. Firstly, RefineNet provides a mean value around five percent lower than the other networks. The Wilcoxon test confirms this observation. Therefore, RefineNet is not recommended for the corrosion detection task. IoU scores of the other three models are very similar in Figure 6.1(a), So the Wilcoxon test is used to further compare these models. The test shows that DeepLab outperforms U-Net while the results for DeepLab and PSPNet are very similar. In order to further evaluate these models, precision and recall scores were computed as discussed in Section 6.2.2.

6.1.2 Precision and Recall Scores

As discussed in Section 5.1 if the user wants to make sure as much corrosion as possible is detected, recall scores should be compared. Figures 6.1(b) and (c) provide the boxplot charts for precision and recall respectively. From these figures, it is evident that the subdivided dataset leads to better precision and recall compared to the resized dataset for both U-Net and RefineNet. So, a resized dataset is not recommended for these two networks. For DeepLab and PSPNet, the results on resized datasets have a lower mean recall score but higher mean precision score. This means that when these two networks are tested on resized dataset, the algorithm tends to predict less corrosion to keep the quality of prediction is high. Therefore, if the user wants to make sure no corrosion is missed, the subdivided dataset is rcommended to be used for DeepLab and PSPNet.

Considering recall scores, PSPNet has a mean value much higher than the other three networks for the subdivided dataset. The Wilcoxon test confirmed this ob-

servation. Therefore, if the user intends to perform a more conservative inspection, PSPNet should be used.

6.1.3 Effect of Batch Size on the Performance of DeepLab

For training the DeepLab V3+ model, it is recommended to use a batch size larger than 12 [77]. Batch size means the number of images to be trained at one time. Larger batch size requires larger GPU memory to process the images. There are four GPUs on the server, and the allowable GPU memory of each GPU device is 12 gigabytes (GB). In order to set the batch size to 12, all four GPU devices need to run at the same time for training. If the batch size is set to 8, only two GPU devices are needed. Since four GPUs are not always available for training, an experiment was performed to test how the model is affected by the batch size used during training. The experiment was performed on a subdivided dataset. All the training parameters were set the same, and only batch size was changed from 12 to 8. IOU scores of test results were calculated. The results show that when the training batch size decreased from 12 to 8, the IOU score also decreased from 0.87 to 0.67. It can be concluded that training the models using a larger batch size leads to a better result.

6.1.4 Cost of Computation of Each Model

The time required to process each 512×512 image, the number of parameters in each model, and GPU memory required to run the test are summarized in Table 6.1. Since test time listed for subdivided dataset is for an image with the size of 512×512, the time required to test the whole image would depend on how many 512×512 images the original image is subdivided into. The time required to perform the test is very different for the four models. U-Net takes the least time to perform the detection task. PSPNet uses 2.2 seconds to process a single image which is relatively slow. Furthermore, the standard deviation of the test time is small enough that can be ignored. All the models have a very similar number of parameters. Only RefineNet

has a number of parameters that is around twice that of other models. All of the models require large GPU memory to perform the test. PSPNet requires the least GPU memory.

Table 6.1.: Average values of time in seconds required to test each 512×512 image, the number of parameters in million (M), and GPU Memory in GB required for testing

	U-Net	**DeepLab**	**PSPNet**	**RefineNet**
Test Time (s)	0.062	0.28	2.2	0.49
Number of Parameters (M)	31	29	37	67
Memory (GB)	12	12	7	9.6

6.2 Samples of Segmentation Results

Three sample test results obtained by different models used in this study that are trained on subdivided or resized datasets are shown in Figures 6.2, 6.3, 6.4, and 6.5. In order to visualize the results, the sample outputs were converted to black and white image, with white pixels (i.e., 255) representing corrosion and black pixels (i.e., 0) representing background.

6.2.1 Results of U-Net

While visualizing results produced by U-Net, it was found that the pixels detected as corrosion had different values of gray scale. A possible reason for this kind of result is that the outputs of the model represent the possibilities of pixels to be corrosion. In order to get the results that could be evaluated a threshold was set so that only the pixels with the gray scale values greater or equal to the threshold would be defined as corrosion. In order to find the value of the threshold that could best identify the corrosion, four thresholds were tested on both subdivided and resized datasets. These four thresholds were chosen based on the percentage of the gray scale: 0.3, 0.4, 0.5,

Table 6.2.: IOU, precision, and recall scores of U-Net model, when different thresholds were used to segment the predicted results

	Crop				**Resize**			
Threshold	77	102	128	153	77	102	128	153
IOU	0.68	0.72	0.66	0.71	0.62	0.63	0.62	NA
Precision	0.77	0.80	0.83	0.81	0.69	0.72	0.75	NA
Recall	0.87	0.86	0.77	0.83	0.87	0.84	0.79	NA

and 0.6 of 255, which are 77, 102, 128, and 153, respectively. The results are shown in Table 6.2. NA in the table means no corrosion was detected using the threshold. From the table, it can be observed that for both subdivided and resized datasets, the highest IOU score is obtained when the threshold of 102 is used. Thus, for all results of U-Net, the pixels with grayscale values greater or equal to 102 were classified as corrosion.

Three samples of results are shown in Figures 6.2, 6.3, 6.4, and 6.5. From Figure 6.2, it can be seen that U-Net performs well if the boundary of the corrosion is clear. However, when there are some background pixels having color values similar to corrosion, the network tends to falsely classify them as corrosion. Also, when the boundary of the corrosion is not clear, as the third image shows, U-Net does not lead to an accurate segmentation.

As shown in Figure 6.3, DeepLab performs well on most sets. It only fails to detect parts of corrosion when the corrosion does not have a clear boundary when the resized dataset is used.

As shown in Figure 6.4, PSPNet also makes good predictions. The problem is that for the subdivided dataset, when there is no corrosion in the sub-image, PSPNet tends to classify anything with a bounday as corrosion. For example, in the second image in Figure 6.4, PSPNet detected part of the sky as corrosion.

Figure 6.5 shows that RefineNet performs the worst. Especially on the resized dataset where it fails to generate detailed predictions.

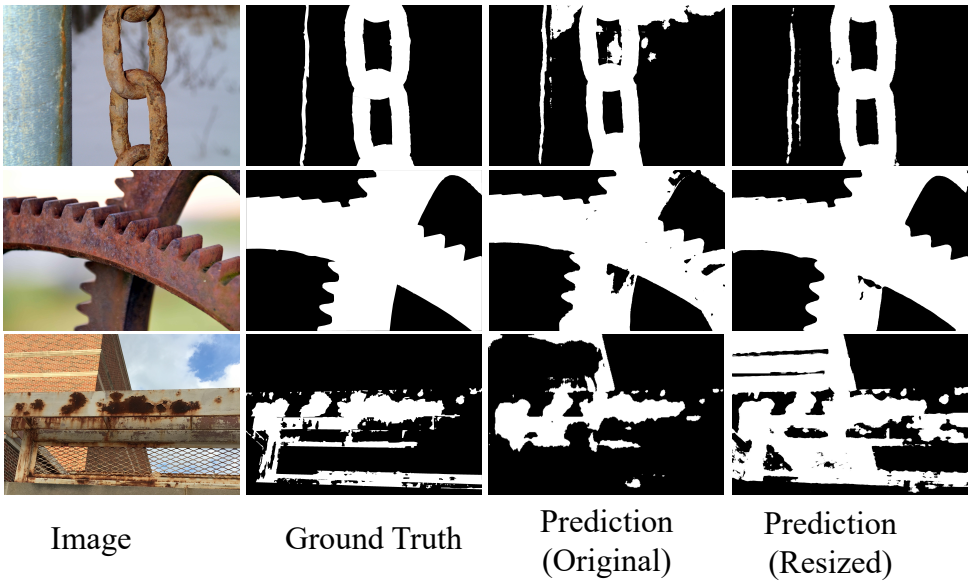


Figure 6.2.: Sample results of U-Net.

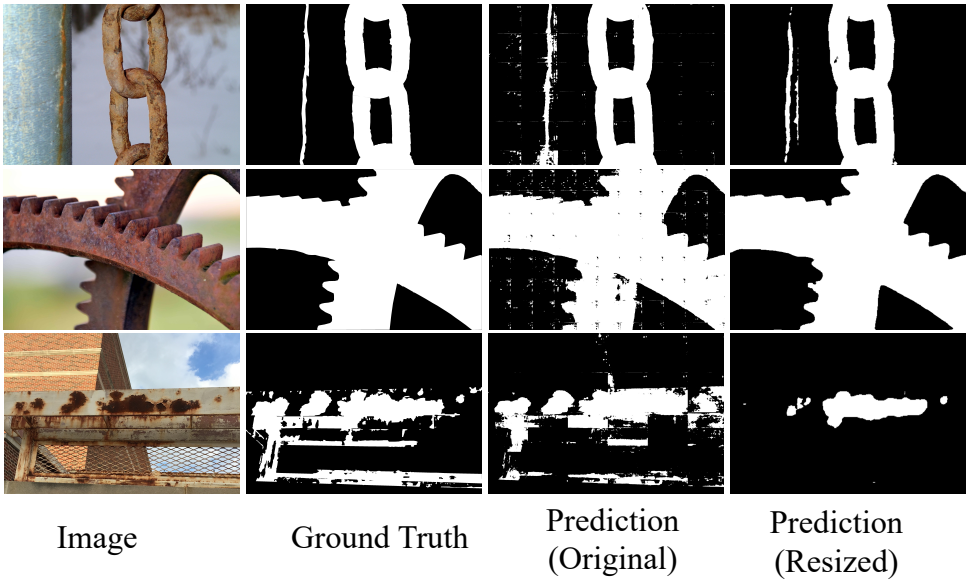


Figure 6.3.: Sample results of DeepLab.

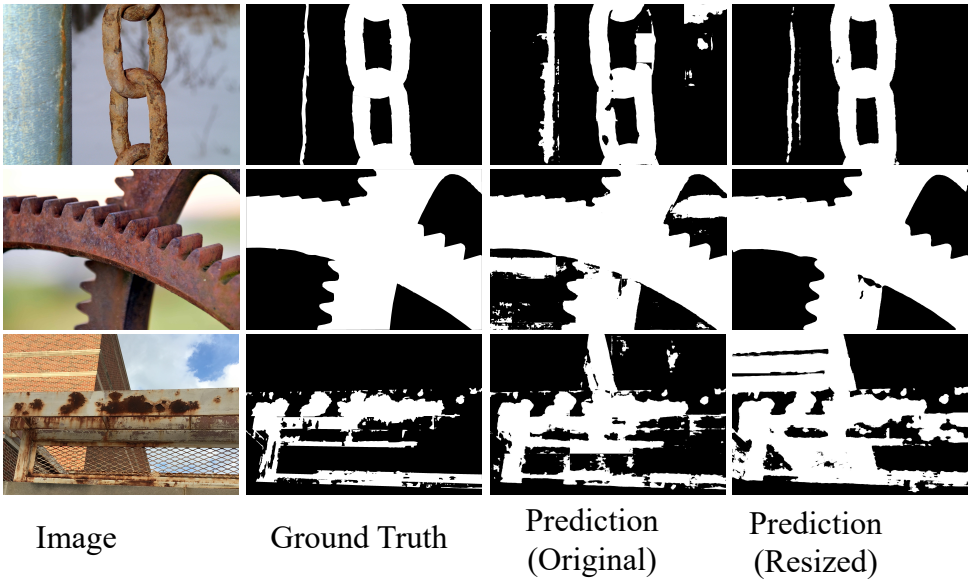


Figure 6.4.: Sample results of PSPNet.

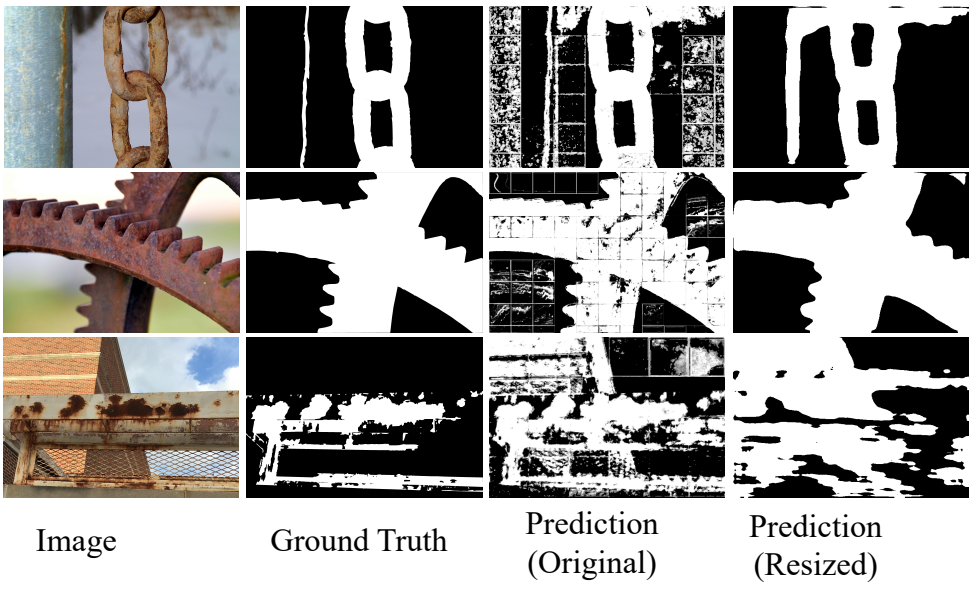


Figure 6.5.: Sample results of RefineNet.

7. CONCLUSIONS AND FUTURE WORK

The objective of this study was to evaluate the performance of four state-of-the-art deep learning-based semantic segmentation models, U-Net, DeepLab, PSPNet, and RefineNet, on corrosion assessment, a critical task in civil engineering. Cross-validation on ten sets of high-resolution digital images was performed for each network for statistical analysis where two cases of subdivided images versus resized (i.e., down-sampled) images of 512×512 where used to this end. Performance evaluation of the models was based on the IoU, precision, and recall scores. Box plots and Wilcoxon signed-rank test were used to analyze whether there is a significant difference between the performance of different models or not.

The results showed that the subdivided dataset provides better accuracy and robustness, and resized dataset enables the prediction in a shorter time. Based on statistical analyses, PSPNet is recommended if the subdivided images are used and it is found that this model is conservative for corrosion detection. DeepLab makes as good predictions on the resized dataset as on the subdivided dataset. Although U-Net could not achieve equally good results, it predicts the results very quickly which can be used for real-time assessments. It is found that RefineNet is not appropriate for corrosion assessment tasks. Regarding the cost of computation, PSPNet requires a long time to process each image but requires the least GPU memory to perform the corrosion detection task. Furthermore, RefineNet has twice the number of parameters in the network than other models which makes it inefficient to be incorporated into robotic systems.

Future work need to focus on improving the efficiency of algorithms to be incorporated into robotic systems and, hence, help engineers make better-informed decisions regarding the maintenance and rehabilitation processes. A depth sensor can be incorporated into the system to quantify the corroded regions in unit area (i.e., in^2)

instead of pixels [78–82]. Additionally, the regions of interest (e.g., bolts) need to be identified so that the robotic system can focus on corrosion detection in critical regions. In order to improve the corrosion detection accuracy, a large number of labeled images need to be developed to enable the development of deep learning solutions and architectures from scratch which can lead to more efficient algorithms. Network pruning [83], which deletes the least important weights, needs special attention for more efficient computations.

REFERENCES

REFERENCES

[1] Gerhardus Koch, Jeff Varney, Neil Thompson, Oliver Moghissi, Melissa Gould, and Joe Payer. International measures of prevention, application, and economics of corrosion technologies study. *NACE International IMPACT Report*, 2016.

[2] AISE. Structural steel: An industry overview. *American Institute of Steel Construction*, 2018.

[3] NBI. Annual materials report on new bridge construction and bridge rehabilitation 2009-2010. *National Bridge Inventory*, 2011.

[4] ASCE. 2017 infrastructure report card. ASCE Reston, VA, 2017.

[5] Gerhardus H Koch, Michiel PH Brongers, Neil G Thompson, Y Paul Virmani, Joe H Payer, et al. Corrosion cost and preventive strategies in the united states. Technical report, United States. Federal Highway Administration, 2002.

[6] American Association of State Highway and Transportation Official. *AASHTO The Manual for Bridge Evaluation*, 2018.

[7] JM Kulicki, Z Prucz, DF Sorgenfrei, DR Mertz, and WT Young. *Guidelines for evaluating corrosion effects in existing steel bridges*. Number 333. 1990.

[8] Luis M Moran Yanez. Bridge maintenance to enhance corrosion resistance and performance of steel girder bridges. 2016.

[9] The Federal Highway Administration of The United States Department of Transportation. *National Bridge Inspection Standard*, 2014.

[10] Elisa Bertino and Mohammad R Jahanshahi. Adaptive and cost-effective collection of high-quality data for critical infrastructure and emergency management in smart citiesframework and challenges. *Journal of Data and Information Quality (JDIQ)*, 10(1):1, 2018.

[11] Jongseong Choi, Chul Yeum, Shirley Dyke, and Mohammad Jahanshahi. Computer-aided approach for rapid post-event visual evaluation of a building façade. *Sensors*, 18(9):3017, 2018.

[12] Muhammad Ali Akbar, Uvais Qidwai, and Mohammad R Jahanshahi. An evaluation of image-based structural health monitoring using integrated unmanned aerial vehicle platform. *Structural Control and Health Monitoring*, 26(1):e2276, 2019.

[13] Mohammad R Jahanshahi, Fu-Chen Chen, Adnan Ansar, Curtis W Padgett, Daniel Clouse, and David S Bayard. Accurate and robust scene reconstruction in the presence of misassociated features for aerial sensing. *Journal of Computing in Civil Engineering*, 31(6):04017056, 2017.

[14] Mohammad R Jahanshahi, Wei-Men Shen, Tarutal Ghosh Mondal, Mohamed Abdelbarr, Sami F Masri, and Uvais A Qidwai. Reconfigurable swarm robots for structural health monitoring: a brief review. *International Journal of Intelligent Robotics and Applications*, 1(3):287–305, 2017.

[15] Mohammad R Jahanshahi, Jonathan S Kelly, Sami F Masri, and Gaurav S Sukhatme. A survey and evaluation of promising approaches for automatic image-based defect detection of bridge structures. *Structure and Infrastructure Engineering*, 5(6):455–486, 2009.

[16] Sanjay Kumar Ahuja and Manoj Kumar Shukla. A survey of computer vision based corrosion detection approaches. In *International Conference on Information and Communication Technology for Intelligent Systems*, pages 55–63. Springer, 2017.

[17] Billie F Spencer Jr, Vedhus Hoskere, and Yasutaka Narazaki. Advances in computer vision-based civil infrastructure inspection and monitoring. *Engineering*, 5(2):199–222, 2019.

[18] Stefan Livens, Paul Scheunders, Gert Van de Wouwer, Dirk Van Dyck, Hilde Smets, Johan Winkelmans, and Walter Bogaerts. A texture analysis approach to corrosion image classification. *Microscopy Microanalysis Microstructures*, 7(2):143–152, 1996.

[19] Ramana M Pidaparti, Babak Seyed Aghazadeh, Angela Whitfield, AS Rao, and Gerald P Mercier. Classification of corrosion defects in nial bronze through image analysis. *Corrosion Science*, 52(11):3661–3666, 2010.

[20] P Chun, K Funatani, S Furukwa, and M Ohga. Grade classification of corrosion damage on the surface of weathering steel members by digital image processing. In *Proceedings of the Thirteenth East Asia-Pacific Conference on Structural Engineering and Construction (EASEC-13)*, pages G–4. The Thirteenth East Asia-Pacific Conference on Structural Engineering and , 2013.

[21] Young-Jin Cha, Wooram Choi, Gahyun Suh, Sadegh Mahmoudkhani, and Oral Büyüköztürk. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):731–747, 2018.

[22] Deegan J Atha and Mohammad R Jahanshahi. Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection. *Structural Health Monitoring*, 17(5):1110–1128, 2018.

[23] Mohammad R Jahanshahi, Farrokh Jazizadeh, Sami F Masri, and Burcin Becerik-Gerber. Unsupervised approach for autonomous pavement-defect detection and quantification using an inexpensive depth sensor. *Journal of Computing in Civil Engineering*, 27(6):743–754, 2012.

[24] Mohammad R Jahanshahi and Sami F Masri. Adaptive vision-based crack detection using 3d scene reconstruction for condition assessment of structures. *Automation in Construction*, 22:567–576, 2012.

[25] Mohammad R Jahanshahi and Sami F Masri. A new methodology for non-contact accurate crack width measurement through photogrammetry for automated structural safety evaluation. *Smart materials and structures*, 22(3):035019, 2013.

[26] Mohammad R Jahanshahi, Sami F Masri, Curtis W Padgett, and Gaurav S Sukhatme. An innovative methodology for detection and quantification of cracks through incorporation of depth perception. *Machine vision and applications*, 24(2):227–241, 2013.

[27] Mohammad R Jahanshahi, Fu-Chen Chen, Chris Joffe, and Sami F Masri. Vision-based quantitative assessment of microcracks on reactor internal components of nuclear power plants. *Structure and Infrastructure Engineering*, 13(8):1013–1026, 2017.

[28] Fu-Chen Chen, Mohammad R Jahanshahi, Rih-Teng Wu, and Chris Joffe. A texture-based video processing methodology using bayesian data fusion for autonomous crack detection on metallic surfaces. *Computer-Aided Civil and Infrastructure Engineering*, 32(4):271–287, 2017.

[29] Mohammad R Jahanshahi, Sami F Masri, and Gaurav S Sukhatme. Multi-image stitching and scene reconstruction for evaluating defect evolution in structures. *Structural Health Monitoring*, 10(6):643–657, 2011.

[30] Po-Han Chen and Luh-Maan Chang. Intelligent steel bridge coating assessment using neuro-fuzzy recognition approach. *Computer-Aided Civil and Infrastructure Engineering*, 17(5):307–319, 2002.

[31] S Lee, LM Chang, and PH Chen. Performance comparison of bridge coating defect recognition methods. *Corrosion*, 61(1):12–20, 2005.

[32] Heng-Kuang Shen, Po-Han Chen, and Luh-Maan Chang. Automated steel bridge coating rust defect recognition method based on color and texture feature. *Automation in Construction*, 31:338–356, 2013.

[33] Kuo-Wei Liao and Yi-Ting Lee. Detection of rust defects on steel bridge coatings via digital image recognition. *Automation in Construction*, 71:294–306, 2016.

[34] Julianne Alyson I Diaz, Manuel I Ligeralde, John Anthony C Jose, and Argel A Bandala. Rust detection using image processing via matlab. In *TENCON 2017-2017 IEEE Region 10 Conference*, pages 1327–1331. IEEE, 2017.

[35] Xianghua Xie. A review of recent advances in surface defect detection using texture analysis techniques. *ELCVIA: electronic letters on computer vision and image analysis*, 7(3):1–22, 2008.

[36] Mohammad R Jahanshahi and Sami F Masri. Parametric performance evaluation of wavelet-based corrosion detection algorithms for condition assessment of civil infrastructure systems. *Journal of Computing in Civil Engineering*, 27(4):345–357, 2012.

[37] Robert M Haralick, Karthikeyan Shanmugam, et al. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610–621, 1973.

[38] Francisco Bonnin-Pascual and Alberto Ortiz. Combination of weak classifiers for metallic corrosion detection and guided crack location. In *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, pages 1–4. IEEE, 2010.

[39] Fátima NS Medeiros, Geraldo LB Ramalho, Mariana P Bento, and Luiz CL Medeiros. On the evaluation of texture and color features for nondestructive corrosion detection. *EURASIP Journal on Advances in Signal Processing*, 2010(1):817473, 2010.

[40] Sindhu Ghanta, Tanja Karp, and Sangwook Lee. Wavelet domain detection of rust in steel bridge images. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 1033–1036. IEEE, 2011.

[41] Hyojoo Son, Nahyae Hwang, Changmin Kim, and Changwan Kim. Rapid and automated determination of rusted surface areas of a steel bridge for robotic maintenance systems. *Automation in Construction*, 42:13–24, 2014.

[42] Heng-Kuang Shen, Po-Han Chen, and Luh-Maan Chang. Human-visual-perception-like intensity recognition for color rust images based on artificial neural network. *Automation in Construction*, 90:178–187, 2018.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[44] Ty Nguyen, Tolga Ozaslan, Ian D Miller, James Keller, Giuseppe Loianno, Camillo J Taylor, Daniel D Lee, Vijay Kumar, Joseph H Harwood, and Jennifer Wozencraft. U-net for mav-based penstock inspection: an investigation of focal loss in multi-class segmentation for corrosion identification. *arXiv preprint arXiv:1809.06576*, 2018.

[45] Will Nash, Tom Drummond, and Nick Birbilis. Quantity beats quality for semantic segmentation of corrosion in images. *arXiv preprint arXiv:1807.03138*, 2018.

[46] Vedhus Hoskere, Yasutaka Narazaki, Tu Hoang, and BillieF Spencer Jr. Vision-based structural inspection using multiscale deep convolutional neural networks. *arXiv preprint arXiv:1805.01055*, 2018.

[47] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[48] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[49] Donald F Specht. Probabilistic neural networks. *Neural networks*, 3(1):109–118, 1990.

[50] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[51] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010.

[52] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[53] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[54] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[55] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[56] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[57] Brendan McMahan and Matthew Streeter. Delay-tolerant algorithms for asynchronous distributed online learning. In *Advances in Neural Information Processing Systems*, pages 2915–2923, 2014.

[58] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[59] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv:1512.03385 [cs]*, Dec 2015.

[61] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[62] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[63] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. pages 2881–2890, 2017.

[64] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[65] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. pages 1925–1934, 2017.

[66] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

[67] Yoh-Han Pao, Gwang-Hoon Park, and Dejan J Sobajic. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2):163–180, 1994.

[68] Amanda JC Sharkey. *Combining artificial neural nets: ensemble and modular multi-net systems*. Springer Science & Business Media, 2012.

[69] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[70] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[71] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[72] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[73] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[74] David F Williamson, Robert A Parker, and Juliette S Kendrick. The box plot: a simple visual method to interpret data. *Annals of internal medicine*, 110(11):916–921, 1989.

[75] Edmund A Gehan. A generalized wilcoxon test for comparing arbitrarily singly-censored samples. *Biometrika*, 52(1-2):203–224, 1965.

[76] Roy C. Milton. An extended table of critical values for the mann-whitney (wilcoxon) two-sample statistic. *Journal of the American Statistical Association*, 59(307):925934, 1964.

[77] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[78] Yulu Luke Chen, Mohammad R Jahanshahi, Preetham Manjunatha, WeiPhang Gan, Mohamed Abdelbarr, Sami F Masri, Burcin Becerik-Gerber, and John P Caffrey. Inexpensive multimodal sensor fusion system for autonomous data acquisition of road surface conditions. *IEEE Sensors Journal*, 16(21):7731–7743, 2016.

[79] Yulu Luke Chen, Mohamed Abdelbarr, Mohammad R Jahanshahi, and Sami F Masri. Color and depth data fusion using an rgb-d sensor for inexpensive and contactless dynamic displacement-field measurement. *Structural Control and Health Monitoring*, 24(11):e2000, 2017.

[80] Mohamed Abdelbarr, Yulu Luke Chen, Mohammad R Jahanshahi, Sami F Masri, Wei-Men Shen, and Uvais A Qidwai. 3d dynamic displacement-field measurement for structural health monitoring using inexpensive rgb-d based sensor. *Smart Materials and Structures*, 26(12):125016, 2017.

[81] Sayna Firoozi Yeganeh, Amir Golroo, and Mohammad R Jahanshahi. Automated rutting measurement using an inexpensive rgb-d sensor fusion approach. *Journal of Transportation Engineering, Part B: Pavements*, 145(1):04018061, 2018.

[82] Ahmadreza Mahmoudzadeh, Amir Golroo, Mohammad R Jahanshahi, and Sayna Firoozi Yeganeh. Estimating pavement roughness by fusing color and depth data obtained from an inexpensive rgb-d sensor. *Sensors*, 19(7):1655, 2019.

[83] Rih-Teng Wu, Ankush Singla, Mohammad R Jahanshahi, Elisa Bertino, Bong Jun Ko, and Dinesh Verma. Pruning deep convolutional neural networks for efficient edge computing in condition assessment of infrastructures. *Computer-Aided Civil and Infrastructure Engineering*, 2019.