

University of Nevada, Reno

**Automatic Concrete Defect Identification by Silencing Features of Deep
Neural Network**

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in
Computer Science and Engineering

by

Umme Hafsa Billah

Dr. Hung M. La - Thesis Advisor
Dr. Alireza Tavakkoli- Thesis Co-Advisor
August 2020



THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

Umme Hafsa Billah

entitled

**Automatic Concrete Defect Identification by Silencing
Features of Deep Neural Network**

be accepted in partial fulfillment of the
requirements for the degree of

Master of Science

Hung Manh La, Ph.D.
Advisor

Alireza Tavakkoli, Ph.D.
Co-advisor

Gokhan Pekcan, Ph.D.
Graduate School Representative

David W. Zeh, Ph.D., Dean
Graduate School

August, 2020

Abstract

An autonomous concrete crack inspection system is necessary for preventing hazardous incidents arising from deteriorated concrete surfaces. In this thesis, we represent a concrete crack detection framework to aid the process of automated inspection. Deep neural networks highly suffer from the gradient vanishing problem [1]. The effect of gradient vanishing problem is very prominent on class imbalanced data-sets such as crack detection. In this work, a deep neural architecture is proposed for alleviating the effect gradient vanishing problem. Furthermore, A feature silencing module is incorporated in the crack detection framework, for eliminating unnecessary feature maps from the network. This module reduces the computational costs of deep neural networks. The overall performance of the network significantly improves as a result. Experimental results support the benefit of incorporating feature silencing within a convolutional neural network architecture for improving the network's robustness, sensitivity, and specificity. An added benefit of the proposed architecture is its ability to accommodate for the trade-off between specificity (positive class detection accuracy) and sensitivity (negative class detection accuracy) with respect to the target application. Furthermore, the proposed framework achieves a high precision rate and processing time than crack detection architectures present in literature.

Acknowledgments

I would like to thank my advisor, Dr. Hung La, who provided me with the guidance and assistance needed for my research. I would also like to thank my co-advisor and committee member Dr. Alireza Tavakkoli for his keen supervision and assistance for my research. Furthermore, my thanks goes to Dr. Gokhan Peckan for the thorough review of this work. This work is supported by the U.S. National Science Foundation (NSF) under grants NSF-CAREER: 1846513 and NSF-PFI-TT: 1919127, and the U.S. Department of Transportation, Office of the Assistant Secretary for Research and Technology (USDOT/OST-R) under Grant No. 69A3551747126 through INSPIRE University Transportation Center (<http://inspire-utc.mst.edu>) at Missouri University of Science and Technology, and the Japan NineSigma through the Penta-Ocean Construction Ltd. Co. under Agreement No. SP-1800087. The views, opinions, findings and conclusions reflected in this publication are solely those of the authors and do not represent the official policy or position of the NSF, the USDOT/OST-R and any other entities.

Table of Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 1.1 | Literature Review | 2 |
| 1.2 | Contributions | 8 |
| 1.3 | Thesis Organization | 9 |
| 2 | Background | 10 |
| 2.1 | Convolutional Neural Network Architecture | 10 |
| 2.2 | Image Classification based Architecture | 11 |
| 2.2.1 | CNN architecture proposed by [2] | 11 |
| 2.2.2 | CNN architecture proposed by [3] | 12 |
| 2.2.3 | Crack Detection using Residual Network Architecture (ResNet) [1] | 13 |
| 2.3 | Encoder Decoder Architecture | 14 |
| 2.4 | Drawbacks of Encoder-decoder architectures | 16 |
| 3 | Methodology | 21 |
| 4 | Experiment Results | 28 |
| 4.1 | Data-set Preparation | 28 |
| 4.1.1 | Data-set for crack image classification | 29 |
| 4.1.2 | Data-set for encoder-decoder architecture | 29 |

| | | |
|----------|---|-----------|
| 4.2 | Experimental Setup | 32 |
| 4.2.1 | Experimental Setup for Crack Image Classification | 32 |
| 4.2.2 | Experimental Setup for Encoder Decoder Architecture | 33 |
| 4.3 | Result Analysis | 35 |
| 4.3.1 | Qualitative Comparisons | 35 |
| 4.3.2 | Quantitative Comparisons | 41 |
| 4.4 | Network Complexity Analysis | 49 |
| 5 | Conclusion and Future Work | 52 |
| 5.1 | Conclusion | 52 |
| 5.2 | Future Work | 53 |

List of Tables

| | | |
|-----|--|----|
| 1.1 | A review of the methods for crack detection | 5 |
| 2.1 | The ResNet architecture | 14 |
| 3.1 | FSM from an instance of the Illinois Bridge Data-set: the numbers in FSM represent the i^{th} eliminated feature. The initial $(x, y) = (256, 256)$ for FSS_a and FSS_b . Once the image is passed through multiple en- coders, the image size is reduced to half of the original. The (x, y) value for Encoder2, Encoder3, Encoder4 and Encoder5 are $(128, 128)$, $(64, 64)$, $(32, 32)$ and $(16, 16)$ | 27 |
| 4.1 | Crack detection result on several images using three different processes. The correct or false classification is identified by how many 256×256 blocks are classified as crack class. | 36 |
| 4.2 | Quantitative measures used for evaluating the results of deep network architectures. | 42 |
| 4.3 | Overall Ranking of each method based on quantitative measures. | 42 |
| 4.4 | Ranking of each method based on individual quantitative measures. | 43 |
| 4.5 | Ranked methods based on dependent measures. | 46 |
| 4.6 | Ranked methods based on independent measures. | 47 |

| | | |
|-----|--|----|
| 4.7 | Comparison ANet-FSM architecture with crack detection architectures. The ANet-FSM architecture was trained and tested on the data-set prepared by DeepCrack [4]. | 49 |
| 4.8 | Comparison of network complexity of different encoder decoder based architecture. C : network complexity, N_C : number of convolutional layers N_k : number of kernels in each convolutional layer, K_s : a $m \times n$ dimension kernel | 50 |

List of Figures

| | | |
|-----|--|----|
| 1.1 | Example of crack detection: (a) Original image, (b) Image processing technique (Different of Gaussian) | 3 |
| 2.1 | The workflow of crack block detection using the framework proposed by [1] | 12 |
| 2.2 | An encoder-decoder based crack detection architecture proposed by [5] | 16 |
| 2.3 | Training loss of two different type convolution networks. The training loss was extracted from a 20 epoch window while training. N depicts the start of the window and the loss was plotted for each epoch. Training loss of convolution network with two different kernel size (3×3 and 7×7) are plotted. | 18 |
| 2.4 | The comparison of two neighborhood of crack location from a small neighborhood and large neighborhood. Since crack pixels occur a very small amount of time, the small neighborhood only contains crack pixels. On the other hand, in the large neighborhood, the statistical relationship between crack and non-crack pixels can be captured more appropriately | 19 |
| 3.1 | The proposed network architecture overview. | 22 |

| | | |
|-----|--|----|
| 3.2 | Encoder and Decoder Module of ANet-FSM architecture. Each encoder performs a 7×7 convolution and max-pooling operation. The decoders up-sample this low dimensional feature space into upper dimension using bi-linear interpolation. The feature space decoding is performed by the convolution operation in each decoder. | 22 |
| 3.3 | An example of a feature space of different crack images during an encoding operation. The weak feature responses generate a weight matrix close to zero (approximately). This feature maps are eliminated using equation 3.1 | 23 |
| 3.4 | FSM of the proposed architecture: the silenced feature maps are represented with black colors. | 23 |
| 4.1 | (a) Sample crack image and (b) annotated image from the Illinois Bridge data-set. | 30 |
| 4.2 | The training procedure with data augmentation technique for encoder-decoder based crack identification. | 31 |
| 4.3 | Comparison of image classification architectures with encoder-decoder architecture on three sample images. Columns are the tested images: (a) contains clear vertical crack, (b) contains small crack on bottom, (c) contains a crack at arbitrary orientation. Row 1- Original image; Row 2- Gibbs architecture [3]; Row 3- ANet-FSM architecture. | 37 |
| 4.4 | Comparison of different thresholds on the performance of ANet-FSM on four sample images. Columns are the tested images: (a) contains clear horizontal crack, (b) contains clear vertical crack, (c) contains a crack at arbitrary orientation, (d) non-cracked concrete image. Row 1- Original image; Row 2- ANet-FSM (hi); Row 3- ANet-FSM (low); and Row 4- ANet-FSM (opt). Results will be seen clearer when zoom-in. | 39 |

| | | |
|-----|--|----|
| 4.5 | Comparison of different encoder-decoder based architectures on four sample images. Columns are the tested images: (a) contains non-cracked concrete image, (b) contains clear horizontal crack on top, (c) contains clear vertical crack, (d) contains non-crack concrete image. Row 1- Original image; Row 2- SegNet [6]; Row 3- SegNet-SO [5]; Row 4- InspectionNet; Row 5- ANet-FSM architecture. | 40 |
|-----|--|----|

Chapter 1

Introduction

A fully functional and healthy infrastructure is the heart of the modern transportation system. The main element of these infrastructures is concrete. A mixture of several different types of rocks, limestone, clay, and water is known as concrete. The water in the concrete evaporates over time due to environmental factors and continuous usage. As a result, the concrete surface hardens over time, leading to severe deterioration such as cracking, spalling, abrasion, etc. [7–9]. There are other factors responsible for concrete surface deterioration such as overloaded vehicles, chemical exposure, corrosion with the metals infused in concretes, and improper drying as reported in [1, 10]. Due to severe deterioration of concrete surface infrastructure assets require frequent inspection and repair. Furthermore, unavoidable circumstances such as road accidents may occur because of defected infrastructure. Therefore, a proper civil infrastructure inspection system is essential to avoid unwanted circumstances as well as prevent traffic disruption. Manual inspection has been employed for a long time using heavy and large equipment by civil engineers to assess the structural defects. The time-consuming and labor-intensive nature of this type of inspection system causes traffic disruption. Furthermore, the manual assessment procedure is perilous for humans in

inaccessible regions of civil infrastructures such as under bridge decks and underwater beams. On the contrary, an autonomous civil infrastructure inspection system monitors structural health continuously with the least human intervention [11]. Such an autonomous robotic system can capture data for surface-level visual inspection and defect identification of civil infrastructures [3, 12–14]. To aid the autonomous crack inspection process a defect(crack) detection algorithm is proposed in this thesis.

1.1 Literature Review

Several image processing techniques, such as thresholding [15–19], morphological operations [20–25], edge detection algorithms [26–29] and adaptive binarization techniques [30] for crack detection in civil infrastructure evaluation have been reported in the earlier literature. The defected areas of concrete surface exhibit two important properties. First, these defects do not have any definite shape or pattern. As a result, these defects do not represent any distinct feature properties. As a result, application of statistical estimation becomes difficult for accurate identification. The rare occurrence and aberrant shape of concrete defects such as cracks closely resemble the properties of anomalies. Therefore, concrete crack detection can be contemplated as an anomaly detection problem, in which anomaly detection techniques [31, 32] are applicable.

The aforementioned techniques extract both local and global features for effective concrete crack detection. Although these techniques are computationally inexpensive, they have several disadvantages. These techniques generate unnecessary feature points because of the highly textured nature of concrete images. Removal of these feature points (using median or mean filtering) eliminates significant defect locations. The precondition of selecting a unique set of parameters for different image sets is

a challenging task for these image processing methods. Apart from this, the feature properties of cracks and edges are different from each other. The pixel connectivity of edges is smoother than cracks. Unlike edges, crack pixels are not continuously connected. As a result, traditional image processing techniques fail in crack classification [2, 3, 27]. An example of a crack detection result using the traditional image processing method on a clean concrete image is shown in figure 1.1(b). A Difference of Gaussian filter was applied to the original crack image in figure 1.1(b). It is evident from this figure, this filter identifies a lot of false cracks. These crack locations represent crack like appearance in the concrete image. Apart from this, the pixel connectivity in the original image also affects the detection result.

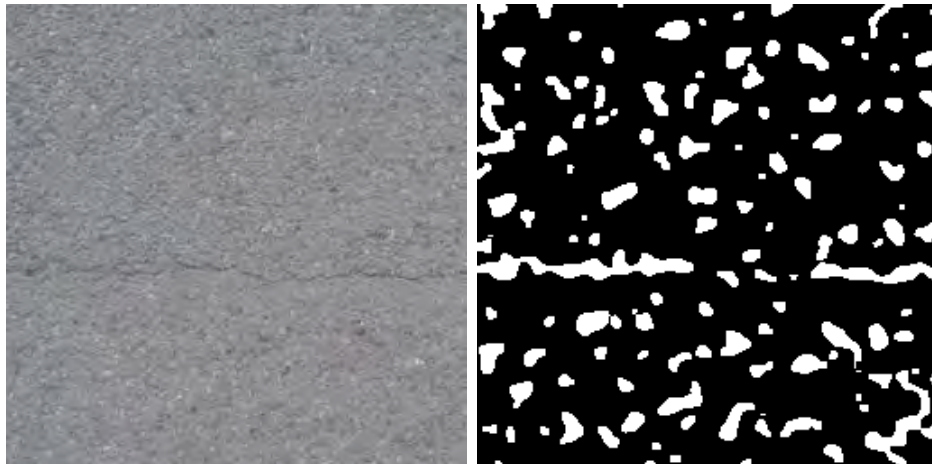


Figure 1.1: Example of crack detection: (a) Original image, (b) Image processing technique (Difference of Gaussian)

Moreover, concrete defect images are affected by environmental non-uniformity such as illumination, noise, shading, and many more. The image processing methods are highly sensitive to these environmental factors [2, 3].

Machine learning architectures were employed for crack detection to achieve robustness toward crack detection. Machine learning architectures such as Support Vector Machines (SVM) [20, 28, 33, 34], Adaboost [28] and Multi-Layer Perceptron (MLP) [35] networks, were used preliminary for concrete crack identification. Later,

a combination of machine learning and image processing techniques were employed to improve the defect detection result [36,37]. Although the accuracy of defect detection improves with these methods, they inherit the complexity of appropriate parameter selection.

Since artificial neural networks (ANN) update the parameter weights autonomously, they are nominally affected by the aforementioned parameter selection problem. Therefore, many ANN structures were employed for concrete distress identification [38,39]. The astounding performance of convolutional neural networks (CNNs) [40] in many image classification and object recognition applications, aspired the researchers to employ CNN architectures for concrete defect identification. CNN architectures closely simulate the functionality of the biological visual cortex by representing cortical areas as individual layers. These layers extract unique feature sets for a specific data-set and learn their statistical properties. Some of the CNN architectures employed for concrete crack identification applications are ResNet [1, 41, 42], AlexNet [43–45], GoogleNet [46] and VGG-Net [47]. Apart from this, CNN architectures were designed specifically for concrete crack identification purpose [2, 3, 48–50].

Table 1.1: A review of the methods for crack detection

| Area | Applied Techniques | Benefits | Drawbacks |
|--------------------------------------|---|---|--|
| Image Processing | Edge Detection operators [26-29] | Useful for extracting both local and global feature points for crack pixel localization | 1. Selection of a perfect parameter set 2. Highly effected by the highly textured concrete background |
| | Thresholding operations [15-19] | | |
| | Morphological operations [20-25] | | |
| | Image Binarization [30] | Introduced an adaptive thresholding for parameter selection | The effect of background noise still remained |
| Machine Learning | Adaboost [28] | Less effected by background noise in comparison to image processing techniques | Require image processing methods for localization |
| | SVM [20, 28, 33, 34] | | |
| | MLP [35], ANN [38,39] | Solves the problem of crack localization | Inherits the complexity of appropriate parameter selection while localizing the cracks |
| | Two stage identification with machine learning and image processing [36,37] | | |
| Deep Learning (Image Classification) | AlexNet [43-45] | 1. Achieves astounding performance for crack image identification. 2. Minimally effected by background noise | Subject to parameter degradation problem. As a result, acheives a high error rate |
| | GoogleNet [46] | | |
| | ResNet [3, 41, 42] | Alleviate the parameter degradation problem | Computationally expensive |
| | Shallow CNN [1, 2, 50] | Computationally inexpensive method | All of these methods require an image processing based localization technique |
| Deep Learning (Encoder-decoder) | ConvNet [59,60] | Acheived significant accuracy over image classification based mehod | The fully connected layers are computationally expensive. |
| | FPCNet [61] | Improved crack detection accuracy using multiple convolution and fully connected layer | Moreover, multiple convolution operation is responsible for parameter degradation |
| | DeepCrack [4, 66] | | |
| | DenseCrack [63] | A high precision result was acheived using feature fusion in clean images | Feature fusion can bring back false positives for complex background images |
| | Surf based CNN [62] | Useful for crack quantification in different orientation. | Surf is very slow feature extractor for real time applications |
| | InspectionNet [54] | Performed crack identification and quantification using SLAM | Subject to parameter degradation problem |
| | SegNet [5,6] | Alleviated the computational complexity of ConvNet | |
| | SDDNet [55] | 1. Uses a large field of view for crack feature extraction. 2. Identifies crack in very complex background | 1. Computationally expensive 2. Very high processing time |

The aforementioned CNN architectures consider crack identification as an image classification problem. Although these techniques are highly efficient for crack feature extraction, they need to utilize image processing algorithms for identifying the exact location of crack [2, 3]. The image processing technique used for defect localization suffers from the parameter selection problem. Moreover, the computational complexity of deep networks stems in the gradient vanishing problem [42], resulting in a significant performance drop.

Semantic segmentation architectures alleviate the problem of localizing and classifying concrete crack pixels at the same time. There exist several semantic segmentation architectures for scene parsing, object instance segmentation, and many other applications. Among them, FCN [51], Mask-RCNN [52], UNet [53] and SegNet [6] have achieved significant performance in the field of segmentation. The object instance segmentation applications widely employ Mask-RCNN architecture. Scene segmentation and image restoration applications employ the SegNet, UNet, and FCN architectures. These architectures learn important feature attributes of the regions through a series of encoding and decoding operations. SegNet, UNet, and FCN architectures are widely employed for scene segmentation and image regeneration applications. These architectures perform a series of encoding and decoding operations for segmentation. The SegNet architecture outperformed UNet and FCN significantly in terms of accuracy, computational complexity, and memory usage.

On the other hand, the ResNet [43] represented the property of extracting efficient features in complex image classification applications. ResNet architecture can be employed as an encoder-decoder manner for segmentation purposes. This approach involves a huge number of parameters, which results in the gradient vanishing problem. Apart from this, the fully connected layer used in ResNet, Mask-RCNN, UNet, FCN architectures increase the computational complexity. The decoder net-

work introduced by the SegNet architecture eliminates the need for using the computationally expensive fully connected layers. Furthermore, excessive computations can lose important crack feature attributes in deeper layers (through max-pooling). The effectiveness of the aforementioned architectures is greatly hampered in that case.

To alleviate the aforementioned problem of crack segmentation, a very few encoder-decoder architectures were proposed in [4, 5, 54–63]. These architectures represented more precise and robust performance than the state-of-the-art semantic segmentation networks.

The classification-based CNN architectures for concrete crack identification classifies a sub-block of an image as crack or non-crack. For example, the network architectures proposed by [2, 3, 49] divide a large image into smaller sub-blocks of size 256×256 . Each of these sub-blocks is a combination of 65536 crack and non-crack pixels. If a crack block contains only 100 crack pixels, the remaining pixels are falsely classified (empirical evidence is shown in chapter 4). Although the false classification rate was alleviated with architectures based on encoder-decoder models proposed by [4, 5, 54–58], these methods significantly suffer from gradient vanishing problem. Additionally, these methods are highly sensitive to environmental non-uniformity. The proposed architecture in this thesis significantly reduces the false classification rate of image classification methods as well as alleviates the drawbacks of encoder-decoder based architectures. In the following chapters, we elaborately discuss the proposed architecture and provide a comparative analysis of different existing architecture for crack detection.

There are certain drawbacks of encoder-decoder architectures (both crack detection and semantic segmentation) that impede crack detection performance. First, the computations performed by these architectures are twice of a regular CNN architecture. These sheer amount of computations enhances the network’s sensitivity

to gradient vanishing problem as well as environmental non-uniformity. Extracting discernible feature sets from a highly imbalanced crack detection data-set is a challenging task. Therefore, in this thesis, we proposed a CNN architecture addressing the aforementioned drawbacks of existing literature. A summary of different methods employed for crack detection and their drawbacks are shown in table 1.1. An extensive analysis and survey of different methods in crack detection is reported in [64].

1.2 Contributions

The main contributions of this thesis are represented as follows:

- In this thesis, we propose an efficient framework for class imbalanced data-sets such as crack detection. The proposed framework incorporates an encoder-decoder network architecture for reducing the effect of gradient vanishing problem. The network architecture (referred to as ANet-FSM) is elaborately discussed in chapter 3. Additionally, our investigation reveals that using this specific type of architecture is effective in applications suffering from high degrees of class imbalance –crack identification (non-crack pixels are much more than crack pixels). Empirical evidence supporting the propositions in this work is represented in chapter 4.
- The second main contribution of this study is the incorporation of a feature silencing module (FSM). The FSM module alleviates the sensitivity of the deep architectures toward feature maps that do not contribute effectively to the optimization of the network loss. The incorporation of the FSM with the proposed CNN architecture significantly reduced the false classification rate of the concrete crack detection process. We have explained the functionality of the FSM

elaborately in chapter 3. The efficiency of the FSM on concrete crack identification data-set is represented in chapter 4.

1.3 Thesis Organization

The thesis is organized as follows: In chapter 2, a different type of CNN architectures for crack detection are discussed. The proposed crack detection framework and its functionality are discussed in chapter 3. This is followed by the elaborate discussion of experimental results, data-set preparation in chapter 4. Lastly, the conclusion of this thesis is drawn in chapter 5.

Chapter 2

Background

2.1 Convolutional Neural Network Architecture

CNN [40] architectures are a composition of various connected layers such as input, convolution, pooling, activation, and output layers. Some auxiliary layers such as batch normalization and drop out layer are also associated according to application purpose. This arrangement of various layers and their functionality extracts discernible feature sets for each object on CNN. Among all the layers of CNN, the convolutional layer is primarily responsible for the salient feature extraction of a specific object. Each convolutional layer performs multiple convolution operations using receptive fields (kernels) of fixed size and variable weights. A unique feature set is generated as an outcome of each convolution operation. The feature maps are generated by all the different weighted kernels from a feature space. The feature space generated by a convolutional layer represents complex non-linear relationships among the input set and output label. To obtain a linear mapping from these non-linear relationships, an activation function is used at the end of the convolutional layers.

Moreover, the output feature space of a convolutional layer is sensitive to a specific feature position. As a result, a down-sampling operation (known as pooling) is performed after each convolutional layer to obtain a position invariant feature space. The pooling operations preserve strong feature responses and eliminates the weak ones using a fixed pooling window size. CNN architectures generate a robust and position invariant feature space through several convolution and pooling operations. A soft-max layer is added at the end of the CNN to assign a normalized probabilistic distribution to the feature responses.

2.2 Image Classification based Architecture

The earlier research of crack identification frameworks classified individual images as crack or non-crack. There have been an extensive amount of research in recent past for crack image classification. Among them, the CNN architectures proposed by [1,3], and [2] achieved significant performance gain over state-of-the-art architectures. These image classification methods divide an input image into multiple small sub-blocks. After that, each sub-block is identified by a CNN module as crack or non-crack. The workflow of a crack image classification architecture using ResNet is represented in figure 2.1.

2.2.1 CNN architecture proposed by [2]

The authors of [2] proposed a shallow CNN architecture for crack image classification. They used image processing based localization technique for identifying the crack location. The CNN architecture consists of four convolutional layers, followed by a batch normalization layer and ReLu activation. A max-pooling operation is used at the end of each convolution layer. The convolution layer uses spatial filters of size

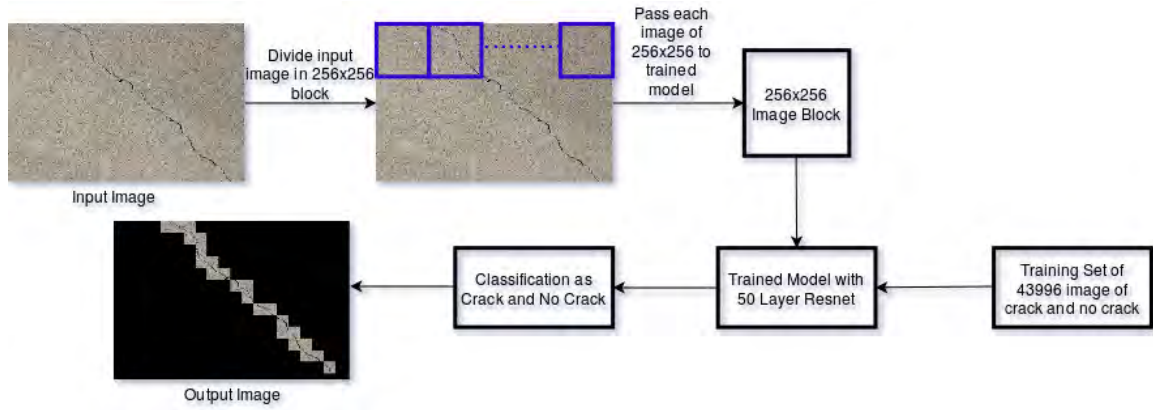


Figure 2.1: The workflow of crack block detection using the framework proposed by [1]

3,24,48 and 96 of size 3 respectively. The weight of these filters is initialized randomly at the start of the training operation. The max-pooling operation downsamples the feature space into a smaller dimension using a 2×2 window with stride 2. A fully connected layer, followed by a soft-max layer is incorporated at the end of the network.

2.2.2 CNN architecture proposed by [3]

The authors of [3] employed a genetic algorithm to identify the parameters of a pre-defined CNN. The estimated parameters involve convolution kernel size, number of filters and number of convolution layers. Through the evolution process, the genetic algorithm finds the best network architecture for a certain type of data-set. The network parameters are represented as a 14-bit chromosome. For each chromosome, a network is constructed and trained on the crack data-set. The fitness function of the genetic algorithm calculates the accuracy of the network after training.

2.2.3 Crack Detection using Residual Network Architecture (ResNet) [1]

The ResNet architecture consists of a fifty layer deep network with residual learning function. The input layer takes an image of a size of $256 \times 256 \times 3$. Those represent the height, width, and number of channels of the image. The input image is passed through a 7×7 convolutional layer, which outputs a 112×112 feature space. A 3×3 max-pooling is performed on the feature space which results in a 52×52 output feature space. The residual function is performed on each three-layer stack consisting of 1×1 , 3×3 , and 1×1 filters. Later, a global average pooling is performed on the network, followed by a 1×1 softmax activation in the fully connected layer.

When extra layers are added to construct a very deep network, it is more efficient to map these layers into a residual function, rather than mapping the added layers to the underlying mapping of the network. Let us assume that the underlying mapping of an added layer is $H(x)$ where x is the input of the first layer. The residual mapping of the added layers would be $F(x) : H(x) - x$. Using the equation $F(x) + x$, a shortcut connection can be added to after several stacked layers, consisting of existing layers and added new layers. This residual learning function is applied to every few stacked layers. Initially, Resnet uses a 32 layer architecture and is applying residual learning in every two layers with 3×3 filters. However, in Resnet with 50 layers a building block is designed with applying a residual function in every three layers. The stacked layers consist of filters with 1×1 , 3×3 , 1×1 dimensions, respectively. The ResNet architecture is shown in table 2.1.

Table 2.1: The ResNet architecture

| Name of the Layer | Output Size | 50 Layer Organization |
|-------------------|-------------|---|
| Conv1 | 112x112 | 7x7, 64, stride 2 |
| Conv2_x | 56x56 | 3x3 max pool, stride 2 |
| | | 1x1, 64 3x3, 64 x 3 1x1, 256 |
| Conv3_x | 28x28 | 1x1, 128 3x3, 128 x 4 1x1, 512 |
| Conv4_x | 14x14 | 1x1, 256 3x3, 256 x 6 1x1, 1024 |
| Conv5_x | 7x7 | 1x1, 512 3x3, 512 x 3 1x1, 2048 |
| | 1x1 | Fully connected layer |

2.3 Encoder Decoder Architecture

Convolutional neural networks have achieved significant performance in applications ranging from object recognition to gesture and pose recognition. In addition to classification and recognition, semantic segmentation and image reconstruction applications employ encoder-decoder based CNN architectures (SegNet [6], UNet [53], FCN [51]). These architectures encode the feature space into a lower dimension through a series of encoders. This feature space is decoded into a higher dimension through corresponding decoders. Salient feature attributes of each pixel of an input data set are learned through these encoding and decoding operations.

For crack identification purpose a number of different encoder-decoder based archi-

tectures were proposed recently such as InspectionNet [54], DeepCrack [4], SDDNet [55] and SegNet-SO [5]. An example of a traditional encoder-decoder architecture (SegNet-SO [5]) is represented in figure 2.2.

The SegNet-SO [5] architecture represented in figure 2.2 is composed of five encoders and decoders. Each encoder combines n number of convolution layers. Two auxiliary layers ReLu and Batch Normalization is employed at the end of the encoder layers. Lastly a max-pooling operation is performed to down-sample the feature space for the next encoding operation. The value of n is set to 3 for the first two encoders. The rest of the encoders set the value of n is to 2.

The decoder layers up-sample an input feature space to its original dimension. The up-sampled feature space passes through the same number of convolution operations as their corresponding encoders. The up-sampling operation was performed using a transposed convolution operation in [5]. The learn-able parameters in transposed convolution enables the network to have more information about the feature space.

Some feature information is lost during the max-pooling operation. This information loss is not trivial for semantic segmentation applications. Nonetheless, this information loss effects the performance of deep network for highly class-imbalance data-sets such as crack data-set. Therefore, a side-output function [65] was employed with each encoder to accumulate for the feature loss. The output feature map of each encoder is up-sampled to original dimension through this operation. The up-sampling operation is performed typically using transposed convolution or bi-linear up-sampling [5]. The up-sampled feature maps from each encoder layers are concatenated together with the output of the final layer. A point-wise convolution layer is used at the end to accumulate all the feature maps to the desired number of classes.

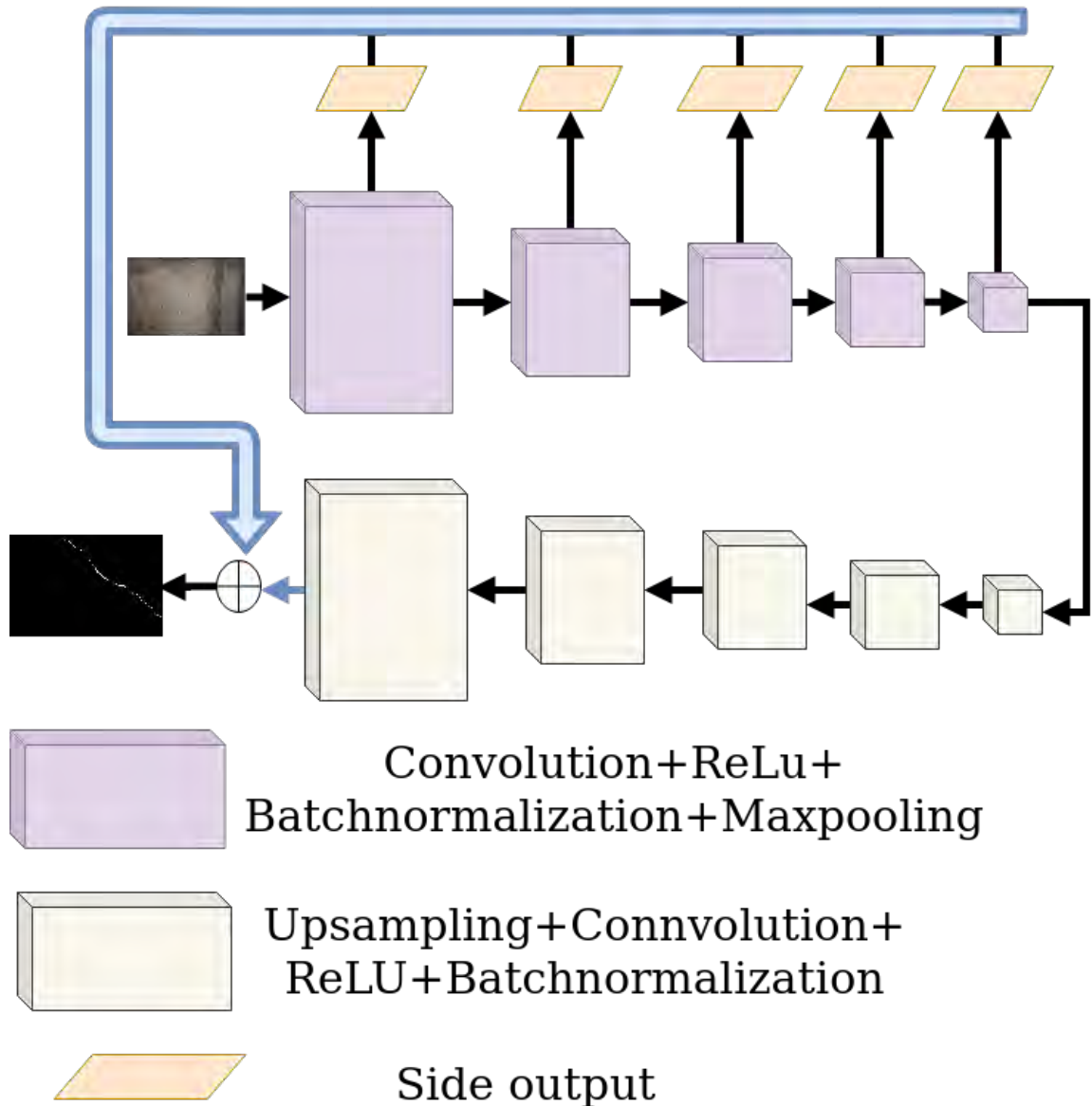


Figure 2.2: An encoder-decoder based crack detection architecture proposed by [5]

2.4 Drawbacks of Encoder-decoder architectures

Although the encoder-decoder architectures represent promising results on crack classification, they have some drawbacks. Firstly, these architectures extract crack feature attributes using multiple 3×3 convolution operation. Multiple convolution operations of small receptive field extract both global and local feature attributes of the input

image as reported in [47]. While going through multiple convolution operations in deeper layers, the gradient stability of a network is lost. The network stops updating the weights of these parameters, which results in the wrong classification of crack pixels. As a result, the network attains a very high loss value. An example of this phenomenon is represented in figure 2.3. In this figure, the loss of two network architectures involving two different convolution operation (3×3 and 7×7) is plotted. We have extracted a 20 epoch window from training for better visualization of the loss. It is evident from the figure that, for multiple 3×3 convolution operation the loss is not stable between any two epochs. The loss doesn't change gradually for this convolution operation. For example, the loss in epoch $N + 1$ jumps to a higher loss in comparison to the loss in epoch N . This phenomenon is visible through the entire 20 epoch window and represents the high gradient instability of this network. On the other hand, for the network with 7×7 kernel size, the loss changes gradually. This represents the stable gradients of the network.

The loss value of both the network in figure 2.3 lies within $\mu \pm std$, where μ is the mean value of the loss in current epoch window and std is the standard deviation. For the network using 3×3 convolution kernel the loss lies within 0.002 ± 0.012 , where $\mu = 0.02$ and $std = 0.012$. The maximum and minimum loss value of this network are 0.05 and 0.002 respectively. On the other hand for the network with 7×7 convolution kernel the loss lies within 0.006 ± 0.0007 , where $\mu = 0.0006$ and $std = 0.0007$. The maximum and minimum loss of this network are 0.0072 and 0.005 respectively. These statistics represent that the loss generalization of a single large convolution kernel (e.g., (7×7)) is better than multiple small kernels (e.g., multiple 3×3 kernel). The loss value of the network with multiple small convolution kernels is seven times (approximately) higher than the network with a large convolution kernel. Due to gradient instability, the network didn't update many parameter weights, resulting

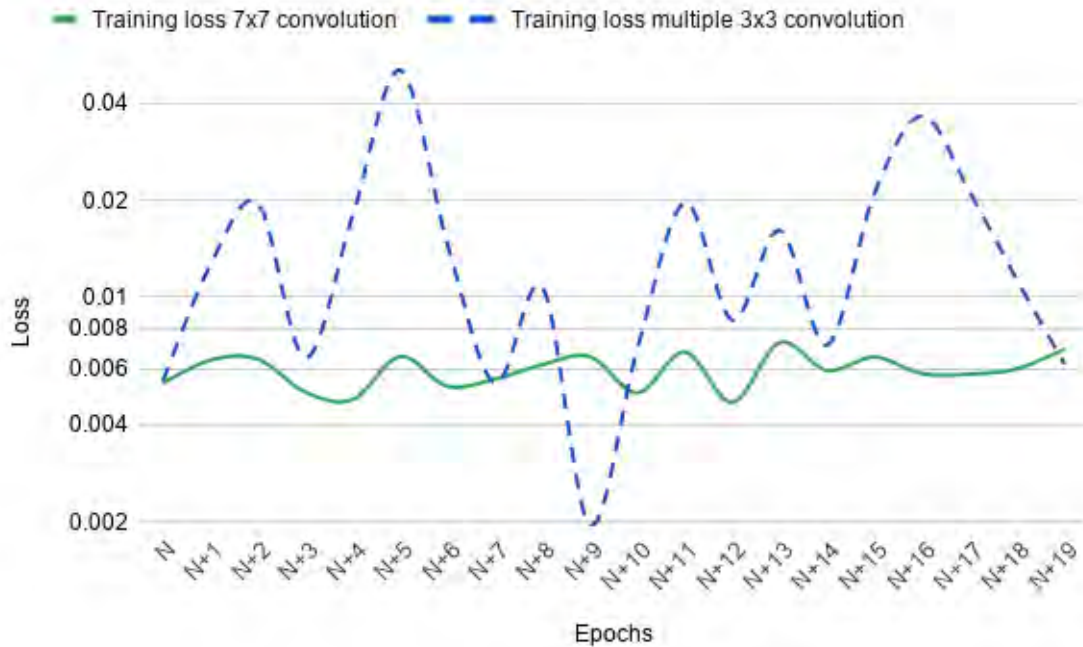


Figure 2.3: Training loss of two different type convolution networks. The training loss was extracted from a 20 epoch window while training. N depicts the start of the window and the loss was plotted for each epoch. Training loss of convolution network with two different kernel size (3×3 and 7×7) are plotted.

in a very high loss value of 0.05. Furthermore, the high standard deviation also represented the highly unstable gradient values or the gradient vanishing problem.

Secondly, the concrete defect identification data sets vary significantly from state-of-the-art image classification databases because crack pixels show anomalous behavior (aberrant patterns and shapes) in comparison to healthy concrete pixels. Additionally, these anomalous pixels appear only in a small portion (2 – 10% approximately) of a concrete image (known as class imbalance problem). Therefore, the relationship between crack and non-crack pixels can be extracted more efficiently if the convolution operation is performed concerning a large neighborhood. An example of the aforementioned phenomenon is represented in figure 2.4. The image in this figure is extracted from the Crack260 [66] Data-Set. The enlarged neighborhood of a crack

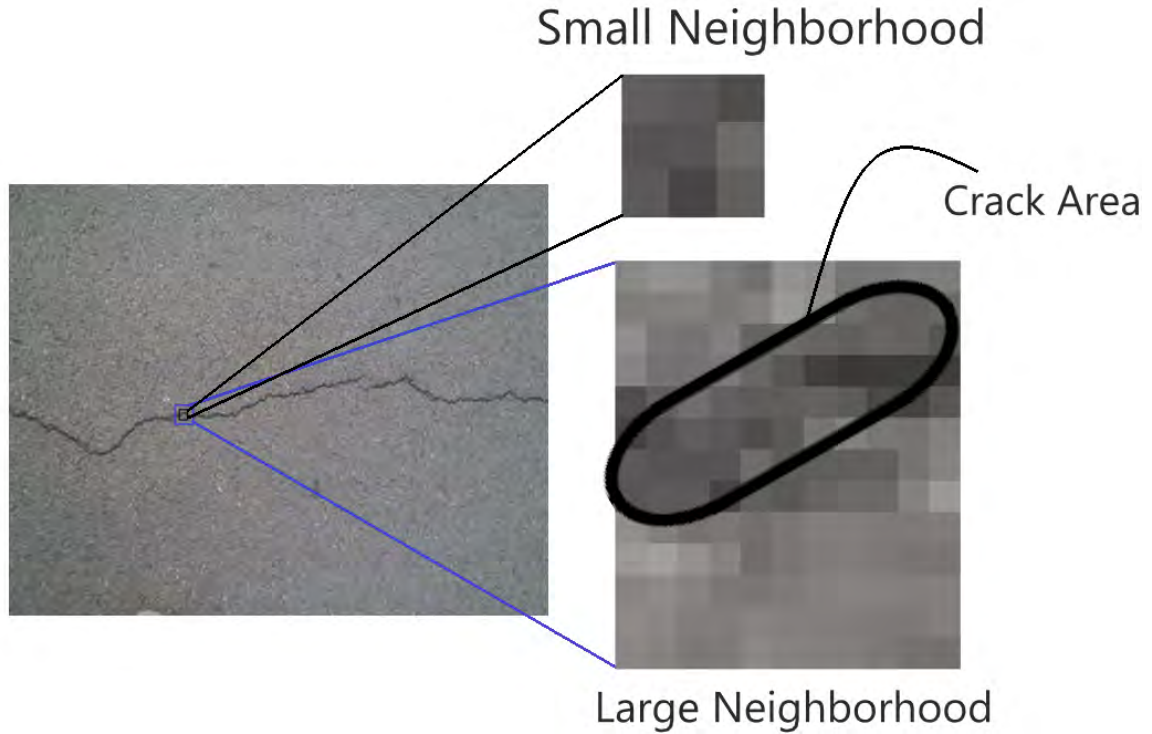


Figure 2.4: The comparison of two neighborhood of crack location from a small neighborhood and large neighborhood. Since crack pixels occur a very small amount of time, the small neighborhood only contains crack pixels. On the other hand, in the large neighborhood, the statistical relationship between crack and non-crack pixels can be captured more appropriately

location is shown in this figure 2.4. It is evident that, with the small spatial neighborhood, we can extract the properties of crack only. Though the global and local features can be extracted using multiple spatial neighborhoods, it is crucial to extract the anomalous relationship between crack and non-crack pixels. This relationship can be extracted using a relatively large neighborhood [55].

For concrete crack classification, the deep crack [4] network architecture achieved a significant performance using multiple receptive fields of size 3×3 . This approach has the advantage of extracting efficient local and global features of crack pixels. As discussed earlier in figure 2.3, the gradient vanishing problem is very prominent in these types of approaches. To address this issue another deep architecture was devel-

oped for crack detection namely SDDNet [55]. This architecture uses a combination of different convolution operations such as DenseSep, atrous and pointwise convolution. They used a large receptive field for extracting the crack and non-crack pixel relationship. In this thesis, we proposed a crack detection framework with only one type of convolution operation. We used a 7×7 convolution kernel with only one convolution layer in each encoding operation. The large receptive field is very crucial for extracting the imbalanced relationship between crack and non-crack pixels as shown in figure 2.4.

Chapter 3

Methodology

The proposed ANet-FSM architecture is comprised of an encoder-decoder module, a feature silencing module (FSM), and a concatenation module. An overview of the whole architecture is shown in figure 3.1. Important crack feature attributes are learned through the encoding and decoding operation.

The FSM eliminates weak feature maps generated from the encoder module and passes the strong feature maps to the concatenation module. The concatenation module up-samples and merges these feature maps together. The up-sampled feature maps along with the output feature map of the decoder module are passed to a 1×1 convolution and soft-max layer. The encoder module in figure 3.2 is composed of five encoder layers.

Each encoder layer assembles a convolutional layer and a max-pooling layer. Two auxiliary layers (Batch Normalization and ReLU) are added at the end of each convolutional layer. The convolution layers in each encoder use 4, 8, 16, 32 and 64 kernels of size 7×7 , respectively. The max-pooling operation down-samples the feature space using a 2×2 pooling window. Additionally, this operation saves the pooling indices in memory to be used in the up-sampling operation in the decoder module.

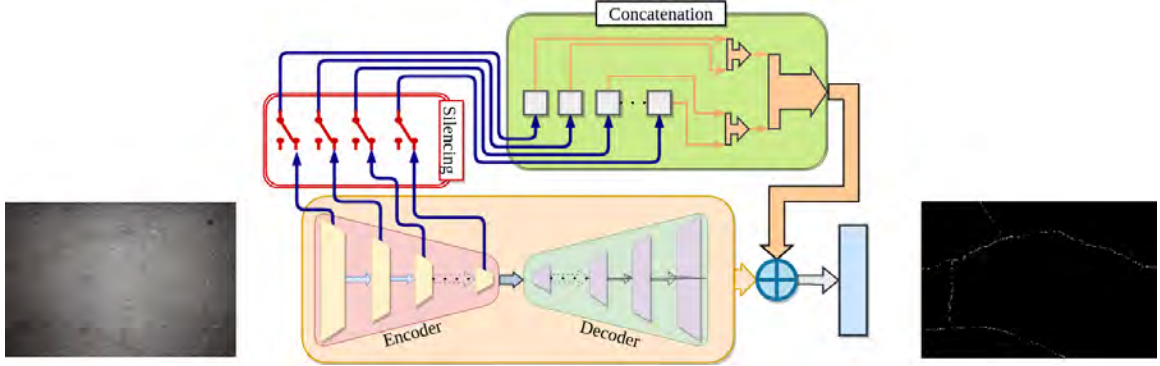


Figure 3.1: The proposed network architecture overview.

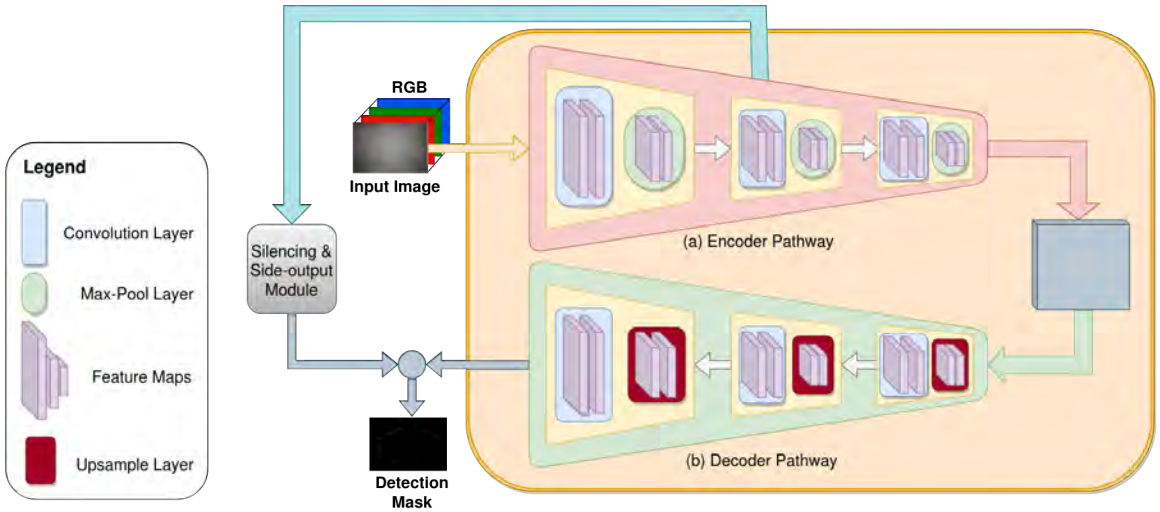


Figure 3.2: Encoder and Decoder Module of ANet-FSM architecture. Each encoder performs a 7×7 convolution and max-pooling operation. The decoders up-sample this low dimensional feature space into upper dimension using bi-linear interpolation. The feature space decoding is performed by the convolution operation in each decoder.

A representation of the decoder layers for each corresponding encoder is represented in figure 3.2. Each decoder layer performs a bi-linear up-sampling operation on its input feature space using the pooling indices saved during max-pooling operation. A convolutional layer is employed at the end of each up-sampling operation to decode the feature space of its corresponding encoder.

A spatial neighborhood of size 7×7 is used for each convolution operation throughout the encoding and decoding operation. As discussed earlier, this neighborhood is

also efficient for eliminating the gradient vanishing problem of state-of-the-art crack detection and semantic segmentation architectures. Therefore, each encoder layer in this architecture is comprised of only one convolutional layer.

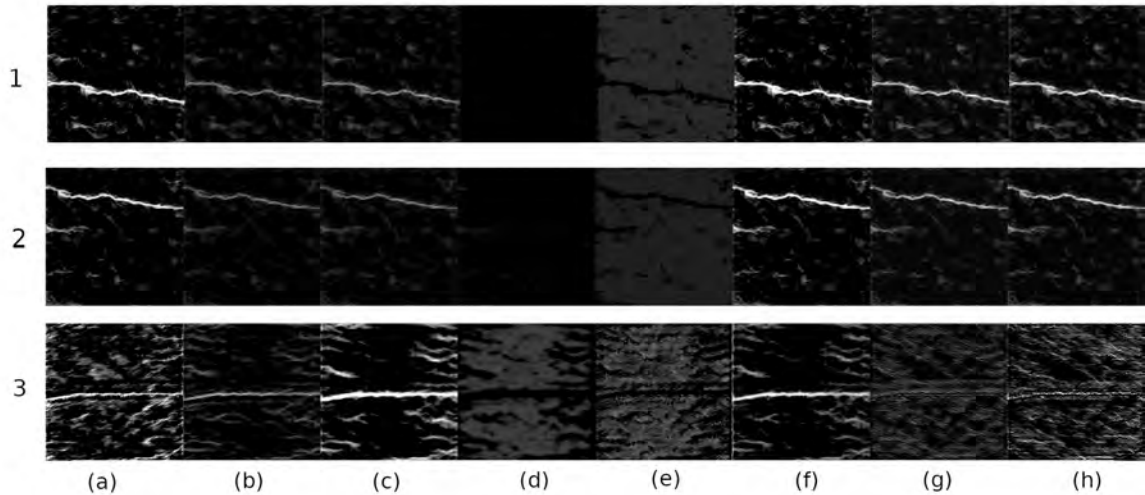


Figure 3.3: An example of a feature space of different crack images during an encoding operation. The weak feature responses generate a weight matrix close to zero (approximately). This feature maps are eliminated using equation 3.1

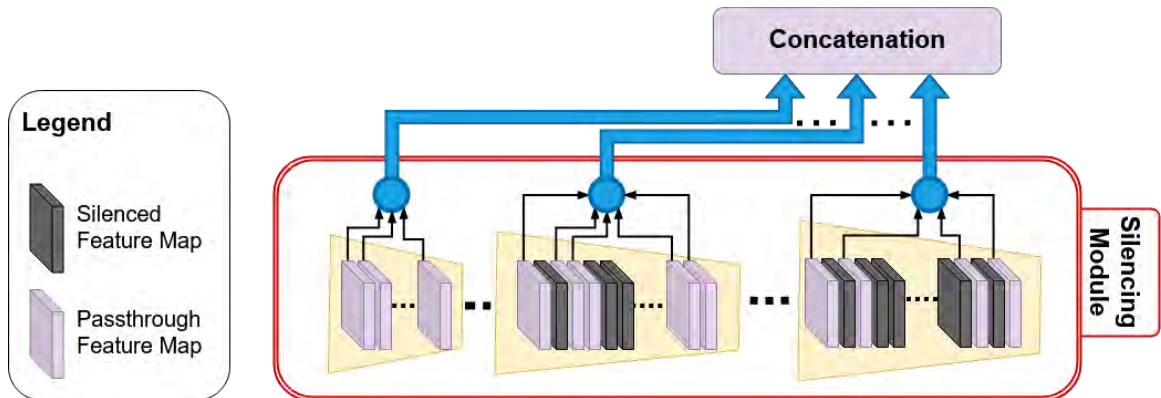


Figure 3.4: FSM of the proposed architecture: the silenced feature maps are represented with black colors.

On the other hand, the max-pooling operation eliminates some feature responses in the course of generating position invariant feature space. This feature loss is nominal when enough instances of different classes are present. The concrete defect data-

sets are highly imbalanced due to the low occurrence of defected pixels. Removal of these feature responses significantly affects the performance of the network. A transient solution to this problem is to up-sample the feature spaces into the original dimension after each encoding operation [5].

Empirical analysis performed on the feature spaces after each encoding operation reveals that all the features in the feature space do not contribute equally to the crack pixel identification. An example of a feature space extracted after a single encoding operation is shown in figure 3.3. It is evident from the figure, the feature maps in figure 3.3(1-3)(a), (b), (c), (f), (h) represent very strong crack features. These feature maps contribute significantly to final crack pixel identification. The remaining feature maps in figure 3.3(1-2)(d), (e), and (3)(g) are responsible for overlapping the salience property of the feature space. Moreover, the weight matrix of these feature space contains gradient values close to zero, which results in false classification. To enhance the precision of crack classification, the FSM module in ANet-FSM architecture eliminates these feature maps from the network feature space. A graphical representation of the FSM module is shown in figure 3.4.

The FSM module in figure 3.4 uses equation 3.1 for extracting significantly contributing feature maps from each encoder feature space. Equation 3.1 is defined as follows:

$$F_p = \bigcup_{f_i \in F}^{i=1 \rightarrow N} f_i \leq th \quad (3.1)$$

$$F_s = F \setminus F_p$$

where F_p is the pruned feature space, F is a feature space from an encoder, f_i is the i^{th} feature map in F , th is a threshold value, F_s is the selected features for passing to the concatenation module and N is the number of feature maps in feature space F .

The encoding operation generates some significantly contributing feature maps for crack detection. Using equation 3.1, we extract these feature maps for up-sampling.

The rest of the feature maps are considered weak by this equation. As a result of gradient vanishing, these feature maps have values close to zero. Hence, in equation 3.1, we select the feature maps for pruning (F_p) having values less than a threshold th . A new feature space F_s is formed by eliminating the feature spaces in F_p from the original feature map F .

The decoder layer in ANet-FSM uses the pooling indices from the max-pooling operation of the corresponding encoder for performing up-sampling (introduced by [6]). The decoding operations are directly dependant on the feature space of the corresponding encoders. Propagating the pruned feature space from one encoder layer to another encoder layer effects the pixel connectivity of the cracks as well as the precision of the network. The feature spaces in encoder and decoder propagate without any pruning operation. Additionally, the pruned feature space (F_s) is passed to the concatenation module shown in 3.1. This concatenation module performs bi-linear up-sampling on the pruned feature space from each encoder (except the first encoder). These up-sampled pruned feature spaces are concatenated with the feature space of the final decoder. Since the crack classification is a one-class classification problem, we use a 1×1 convolution operation at the end of the concatenation operation. This convolution operation helps to estimate the crack location from the concatenated feature space.

An example of eliminated feature maps from an instance of the Illinois Bridge Data-set is represented in table 3.1. To represent the effectiveness of the FSM module in reducing the complexity of a network, we have introduced three measures such as feature silencing rate (FSR), feature space size before pruning (FSS_b) and feature space size after pruning FSS_a . Since there is a scarcity of standardization metrics of feature space pruning, we have incorporated the above three measures for evaluation

only. The FSR in table 3.1 is defined in equation 3.2.

$$FSR = \frac{B_s - A_s}{B_s} \quad (3.2)$$

where B_s represents the number of features before silencing and A_s represents the number of features after silencing. Additionally, we have analyzed the feature space size before and after silencing using equation 3.3.

$$\begin{aligned} FSS_b &= x \times y \times B_s \\ FSS_a &= x \times y \times A_s \end{aligned} \quad (3.3)$$

where x is height of an image, y is width of an image, FSS_b is the the feature space size before silencing and FSS_a is the feature space size after silencing. In table 3.1, we have calculated the value of equation 3.3 for a 256×256 image. Since, each encoder operation reduces the image size to half, the value of x and y is reduced to half. Therefore, image size is $(256, 256)$, $(128, 128)$, $(64, 64)$, $(32, 32)$, $(16, 16)$ for Encoder1, Encoder2, Encoder3, Encoder4 and Encoder5 respectively in table 3.1.

Considering the low number of feature maps generated by the Encoder1, the FSM module does not eliminate feature maps from the feature space generated by this encoder. The feature space from Encoder2 and Encoder3 was reduced to half of their original size. Encoder4 eliminates more than half of the feature maps from feature space. In total, the FSM eliminates 49% of the feature maps generated from the encoder module. Additionally, FSS_a value significantly reduces after silencing for Encoder2, Encoder3, and Encoder4. This represents the effectiveness of the FSM module in reducing the cost of the network. Therefore, we can infer from this phenomenon that almost half of the computations in encoder-decoder based architecture, does not contribute significantly to the prediction. Elimination of such features improve the

Table 3.1: FSM from an instance of the Illinois Bridge Data-set: the numbers in FSM represent the i^{th} eliminated feature. The initial $(x, y) = (256, 256)$ for FSS_a and FSS_b . Once the image is passed through multiple encoders, the image size is reduced to half of the original. The (x, y) value for Encoder2, Encoder3, Encoder4 and Encoder5 are $(128, 128)$, $(64, 64)$, $(32, 32)$ and $(16, 16)$.

| Encoder Number | Features Silencing Rate, FSR | Feature Space Size for 256x256 image | | No. of Features, Fs | | Silenced Feature Set by FSM |
|----------------|------------------------------|--------------------------------------|-----------------------|----------------------|---------------------|-----------------------------|
| | | Before Silencing, FSSb | After Silencing, FSSa | Before Silencing, Bs | After Silencing, As | |
| Encoder1 | 0.0% | 262,144 | 262,144 | 4 | 0 | ♦ |
| Encoder2 | 50.0% | 131,072 | 65,536 | 8 | 4 | ⊖ |
| Encoder3 | 43.8% | 65,536 | 36,864 | 16 | 9 | ↔ |
| Encoder4 | 50.0% | 32,768 | 16,384 | 32 | 16 | ‡ |
| Encoder5 | 53.1% | 16,384 | 7,680 | 64 | 30 | × |
| Encoder Module | 49.0% | | | 124 | 59 | |

♦ = {}

⊖ = {1,4,6,7}

↔ = {1,2,4,5,7,9,12}

‡ = {2,5,6,8,9,10,11,12,13,18,20,21,22, 24,26,29}

× = {1,5,8,11,12,15,17,19,25,26,28-32, 38-44,47-49, 52, 53, 55,56,60, 61-63}

performance as well as reduces the computations of the network remarkably.

Chapter 4

Experiment Results

In this chapter, the results of different CNN architectures on concrete crack identification are compared with the proposed architecture. At first the data-set preparation and augmentation methods are elaborated in section 4.1. Then the experimental set-up for both crack image classification and encoder-decoder architecture is elaborated. Lastly, the results of the proposed architecture are compared with both crack image classification based architecture and encoder-decoder based architecture.

4.1 Data-set Preparation

The experiments in this thesis were performed for both crack image classification and segmentation networks. Since these two types of network require a different type of data-set, we prepared two different data-sets for each type of network. The data-set preparation method is elaborated in the following.

4.1.1 Data-set for crack image classification

The data-set for crack image classification was collected using NDE sensor fusion method represented by [3] from various bridge decks and roads. Furthermore, several published crack images from arbitrary online resources were included in the data-set. These images were divided into 256×256 sub-blocks. Each sub-block was labeled manually into crack and non-crack classes. The data-augmentation operation was performed on this data-set using horizontal or vertical flipping, rotation (left or right) in 90 degree angles. Furthermore, intensity correction operation was performed on the images to incorporate low-contrast images. For this purpose, we have employed the equation, $I = I * \alpha + \beta$, where I is an image pixel, α is set to 1, and β is set to -80. Two different data-sets were created from the original images such as training and validation. The training set includes 21996 images of $256 \times 256 \times 3$ dimensions in the crack class, and 22000 images in the non-crack class. On the other hand, the validation data-set consists of 300 images of crack and non-crack classes. The image classification architectures were tested on 326 images of crack class and 301 images of non-crack class.

4.1.2 Data-set for encoder-decoder architecture

The scarcity of a well-balanced data-set is a challenge for anomalous pixel identification applications such as concrete crack classification. Since the occurrence of crack pixels is much lower than healthy concrete pixels, it is essential to train a deep network architecture with an enormous amount of data instances containing crack pixels. Therefore, we have collected concrete images of different highway bridge decks from different parts of the USA using our previously developed nondestructive evaluation robots [3,14,67] at the Advanced Robotics and Automation Laboratory. We captured images in various light illumination and different times of day and night to incorporate

the non-uniformity of the environment as much as possible in our data-set (referred to as Illinois Bridge data-set). The data-set contains 46 images, where each image has a resolution of 5000×3000 .

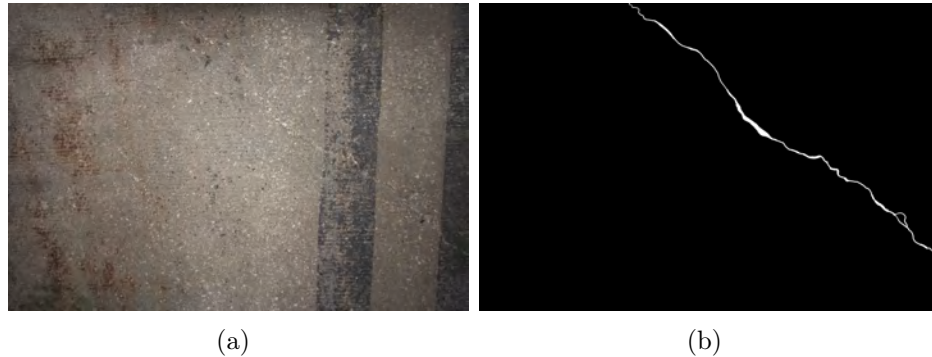


Figure 4.1: (a) Sample crack image and (b) annotated image from the Illinois Bridge data-set.

Appropriate annotation of the images in the data-set is very important for training and validating CNNs. This annotation process is arduous due to the difficulty involved in finding the crack pixels in an image (considering their low occurrence). As a result, the pixels were color-coded for each image individually from the Illinois Bridge data-set. The crack pixels were assigned a white color and the healthy concrete pixels were assigned a black color. An example of our annotated image data-set is shown in figure 4.1.

There exists a number of published data-sets such as Crack260 [66], CrackForest [68]. These data-sets consist of very low resolution images (256×256 , 512×512) in comparison to Illinois Bridge data-set. Additionally, the Illinois Bridge data-set consists of images with several types of noise such as vegetation, road paint, oil spilling, and many more. An example of the annotated data set is shown in figure 4.1(a).

The crack inspection data-sets represents high-class imbalance property. To extract appropriate crack features, it is essential to provide the network with enough

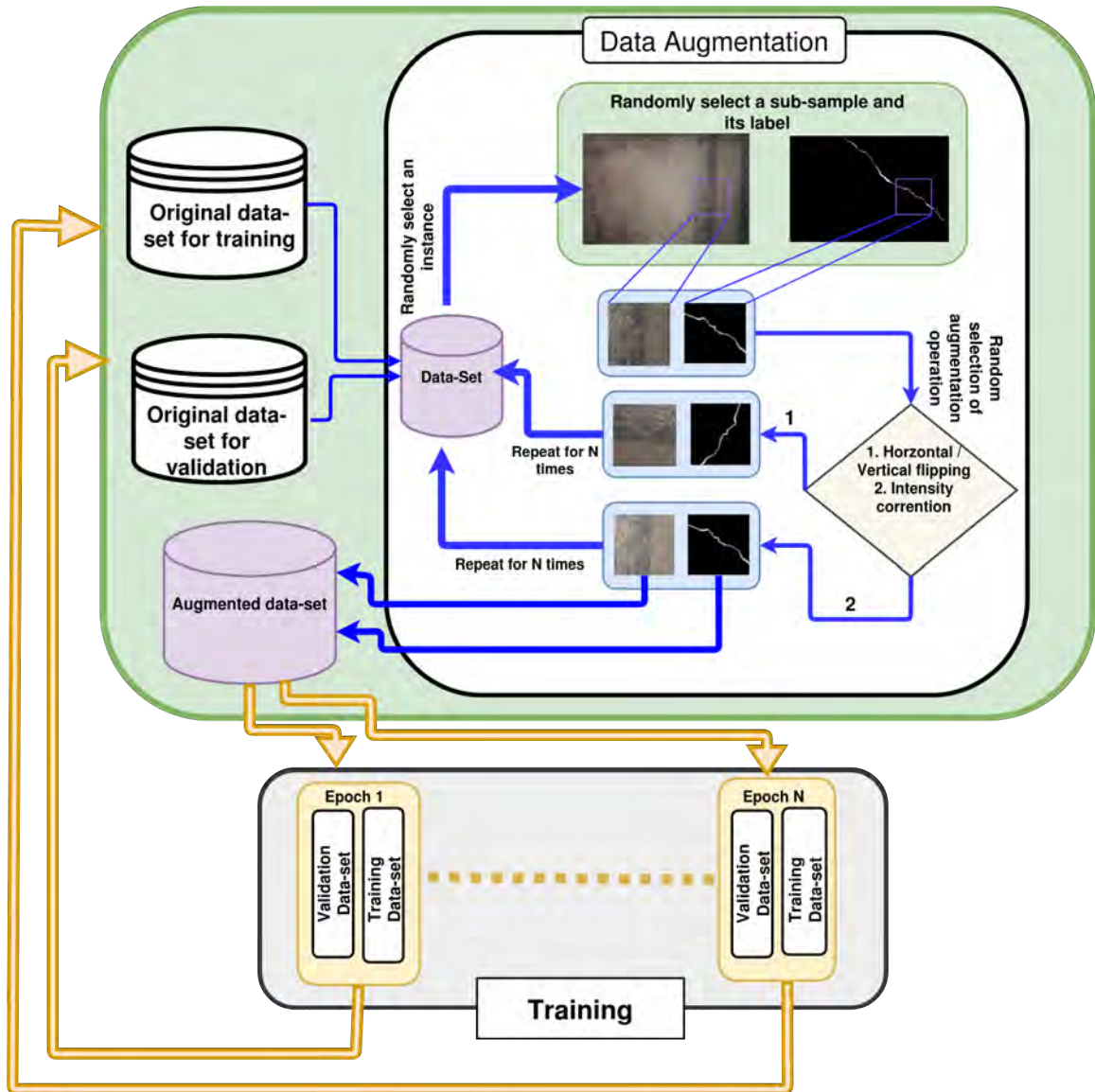


Figure 4.2: The training procedure with data augmentation technique for encoder-decoder based crack identification.

instances of crack samples. Therefore, we employed a data augmentation technique proposed by [5] for this purpose on our original data-set.

The data augmentation technique randomly selects an image from the original data-set (training or validation). A random sub-sample of the selected image is chosen for the data-augmentation operation. This operation is chosen randomly for each individual sub-samples. The data-augmentation operations used in this work are

horizontal flipping, vertical flipping, and gamma correction of intensity. The gamma value is also chosen randomly for intensity correction. This process is repeated for N times, where N represents the size of the data-set. As a result, this technique can generate N sub-samples of augmented data-set from the original high-resolution data-set. A graphical representation of the data augmentation technique is shown in figure 4.2.

4.2 Experimental Setup

The experiments of this thesis are carried on the different types of CNN architectures. An empirical analysis of different crack image classification networks was performed. Then, a comparison of encoder-decoder architecture and image-classification based architecture is performed. Lastly, the proposed architecture is compared with state-of-the-art encoder-decoder architectures in literature. Since the experiments involve two different types of CNN architecture, the experimental setup is different for them. Therefore, the experimental setup is discussed in two sub-sections for crack image classification and encoder-decoder network architecture.

4.2.1 Experimental Setup for Crack Image Classification

Three crack image classification network architecture was employed in this thesis ([2], [3] and ResNet-50 [41]). The network parameters were optimized using Stochastic gradient descent (SGD) with a learning rate of 0.01. The networks were trained with 80 epochs, where weights were updated in each epoch. The initial weights were assigned randomly, rather than using transfer learning from ImageNet data-set.

A real-time image of road or bridge deck surface can contain multiple crack types of different orientations. There may be sub-parts of an image where non-cracks exist.

To identify different patches of an image as a crack and non-crack, it is effective to subdivide an image. Therefore, an image of arbitrary size is taken as input, and the image is then divided into sub-images of $256 \times 256 \times 3$ dimensions (height, width, channel). The images are then fed one by one into the CNN. After that, the images are stitched back together after classifying as crack or non-crack.

4.2.2 Experimental Setup for Encoder Decoder Architecture

The encoder-decoder architectures in this thesis were trained on a 1080 Gtx GPU with 10 Gb memory. For hyper-parameter optimization, Adam optimizer was used with a learning rate of 0.0001.

The proposed deep network architecture was pre-trained on the Illinois Bridge data-set for 300 epochs. Many state-of-the-art CNN network training involves transfer learning from existing networks such as VGG-16 [47]. This transfer learning from the existing network has several disadvantages in our case. Firstly, transfer learning from state-of-the-art image processing data-sets (ImageNet) will give the network unnecessary information on various objects. Since crack pixels have a very small amount of occurrence in an image, the transfer learning process may overlap the features of crack. Secondly, the ANet-FSM architecture prunes feature space weights below a threshold (equation 3.1). Transferring weights from another module (or network) initialize the feature space with high weights. As a result, the FSM module cannot find the weak feature maps for pruning. For the aforementioned reasons, the network was pre-trained on the Illinois Bridge data-set instead of performing transfer learning.

The ANet-FSM network was trained for 300 epochs. At each epoch, data-sets for training and validation are generated using the previously discussed data augmentation technique from the respective original training and validation data-set. In figure

4.2, we represented the process of data-augmentation and training visually. Each image of the training and validation data-set generated by the data-augmentation technique has a resolution of 512×512 . The training data-set consists of 5000 images and the validation data-set consists of 1000 images. As a result, a data-set of 6000 images is generated in each epoch in training. In total, the network is trained and validated on 300 different data-sets respectively.

State-of-the-art data augmentation techniques generate augmented data-set from a handful of low-resolution images for crack detection. As a result, training is performed on the same data-set for each epoch. This type of training procedure is highly susceptible to overfitting problems. Nonetheless, the data-augmentation technique and training process incorporated in this thesis generates different training and validation data-set for each epoch at the time of training. As a result, the network sees a range of different types of images. The overfitting problem of traditional training procedures can be alleviated through this training procedure.

We evaluate the performance of the pre-trained network using the test data-set of 200 images. The testing image size (1024×1024) was selected to be larger than the training image size (512×512) to represent the robustness of the network towards noise in large resolution images. Apart from this, the computational complexity involved with large images generates the gradient vanishing problem in a CNN architecture. The effect of the gradient vanishing problem is more evident in large scale images. Therefore, the resolution of test data-set images is twice larger than the train data-set.

The performance of ANet-FSM architecture was compared with two different type of encoder-decoder architectures such as semantic segmentation and crack image detection architecture. The semantic segmentation architectures such as SegNet suffer from gradient vanishing problem as discussed in chapter 2. The effect of this problem is strongly evident in class-imbalanced data-sets (crack detection data-set). This

effect was evaluated by comparing with semantic segmentation architecture (SegNet [6]). Moreover, several crack segmentation architectures were employed for performance comparison such as InspectionNet [54], SegNet-SO [5], Deep crack [4] and SDDNet [55].

4.3 Result Analysis

In this thesis, we evaluate the results of the proposed network architecture on three different criteria such as qualitative analysis, qualitative measurement and network complexity analysis.













4.3.1 Qualitative Comparisons

The qualitative comparison of several crack identification networks is performed in this section. At first, the CNN architectures for crack image classification are compared. Then, the result of the best crack image classification architecture is compared with encoder-decoder architecture. After that the results of proposed ANet-FSM architecture are compared with block detection architecture in figure 4.3. Finally, the results are compared with the state-of-the-art encoder-decoder architectures such as SegNet [6], SegNet-SO [5] and InspectionNet [54]. We color-coded the true positive pixels (correctly detected crack pixels) with red, false positives (missed crack pixels) with blue, and false negatives (pixels incorrectly labeled as crack) with green, respectively. The true negative pixels (correctly labeled non-crack) are represented with their original texture.

Comparison of CNN Architectures for Crack Image Classification

We have compared the results of three different CNN architectures for crack image classification such as [2,3] and ResNet-50. The results of this architectures are shown in table 4.1.

Table 4.1: Crack detection result on several images using three different processes. The correct or false classification is identified by how many 256×256 blocks are classified as crack class.

| Input image | CNN model by [1] | CNN model by [2] | Fifty layer Resnet model |
|--|--|---|--|
|  Image1 |  Correct = 7, Incorrect = 0 |  Correct = 8, Incorrect = 0 |  Correct = 11, Incorrect = 0 |
|  Image2 |  Correct = 18, Incorrect = 0 |  Correct = 26, Incorrect = 2 |  Correct = 23, Incorrect = 0 |
|  Image3 |  Correct = 20, Incorrect = 79 |  Correct = 24, Incorrect = 16 |  Correct = 24, Incorrect = 14 |

The residual learning network in table 4.1 achieved 94% accuracy after testing on a test set of 627 images of crack and non-crack classes. The CNN architecture proposed by [2] and [3] achieved 89.8% and 96 % accuracy respectively (trained with same data set). Besides, six images of $3486 \times 5184 \times 3$ dimensions were taken and fed into the system. The results show that the Resnet model can identify cracks more accurately than [2] and classifies less false positives than the CNN model by [3]. Table 4.1 shows some examples of the crack detection result of Resnet model, CNN model by [3] and [2]. Though the CNN model by [3] detects more crack pixels, it detects more false positives, in comparison to the ResNet model [1]. In some cases, the Resnet model detects more crack pixels as shown in Image1 of Table 2 than the other two CNN structures. Some cases show that the Resnet detects fewer crack pixels than the

CNN model by [3]. However, Resnet outperforms the CNN model by [2] in all cases.

Comparison of Encoder-Decoder and Image Classification Architecture

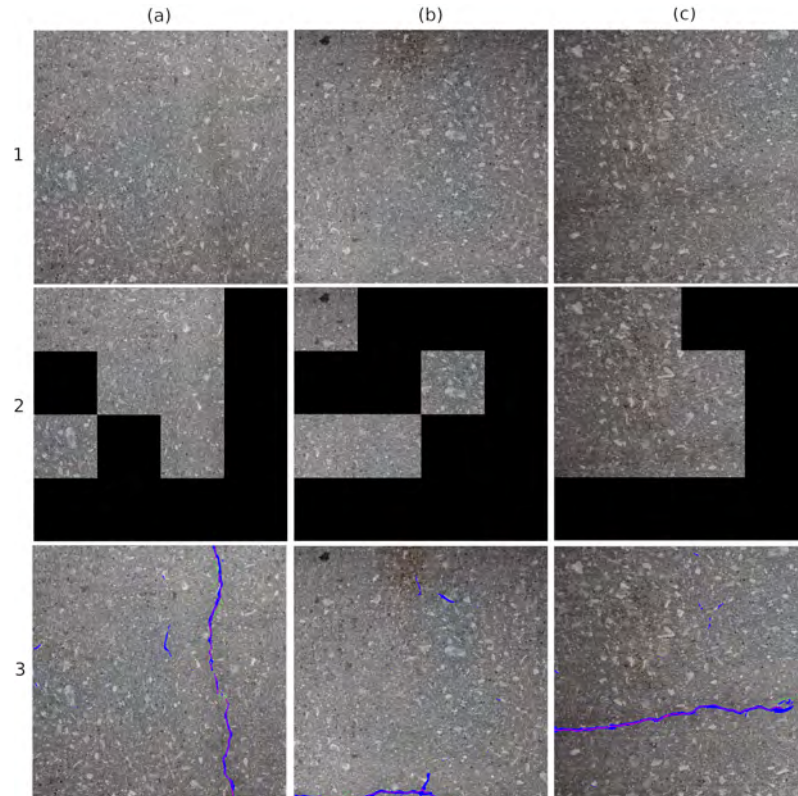


Figure 4.3: Comparison of image classification architectures with encoder-decoder architecture on three sample images. Columns are the tested images: (a) contains clear vertical crack, (b) contains small crack on bottom, (c) contains a crack at arbitrary orientation. Row 1- Original image; Row 2- Gibbs architecture [3]; Row 3- ANet-FSM architecture.

In figure 4.3, the comparative result of the image classification method, and our proposed method are shown. Since the CNN model proposed by Gibbs [3], achieved the highest accuracy in the previous section, we have compares the results of the proposed architecture with this method. The Gibbs [3] architecture, divides an image into smaller sub-blocks of size 256×256 . Each block is identified as crack and non-crack. The non-crack blocks are labeled as black in the image. The results of the

proposed architecture on the same block are shown in figure 4.3, that the crack image classification method identifies an extensive number of non-crack pixels as crack pixels. Moreover, this architecture cannot localize crack pixels at all. On the other hand, the proposed ANet-FSM architecture can classify and localize crack pixels at the same time. For example, in image 2(a) Gibbs method falsely identifies four 256×256 blocks as crack blocks. The ANet-FSM architecture in figure 3(a) represents the exact crack location as well as misidentifies a very less number of crack pixels in comparison to Gibbs architecture.

Comparison of Encoder-Decoder Architectures

We have represented the results of ANet-FSM architecture on three thresholding values such as high (15%), low(10%), and optimal (14%) in figure 4.4. If a low threshold is applied, the network becomes more sensitive towards the environmental non-uniformity and gradient vanishing problem. On the other hand, the network becomes more specific to correct classification. As a result, the number of falsely identified (blue colors) and correctly classified pixels (red colors) of the network significantly increases (shown in row 2(a),(b),(c) of figure 4.4). When a high threshold is applied (shown in row 3(a),(b),(c) of figure 4.4) the sensitivity towards noise is alleviated but the specificity of correct classification also decreases. Application of an optimal threshold not only reduces the false classification rate but also increases correct classifications significantly. This thresholding obtains a balance between specificity and sensitivity as shown in figure 4.4.

The result of different encoder-decoder networks such as SegNet, SegNet-SO, and InspectionNet, and the proposed ANet-FSM architecture is shown in figure 4.5.

The results of 2(a), (c), and (d) in figure 4.5 show that the SegNet architecture has more falsely classified pixels (blue colored) than the remaining networks. The

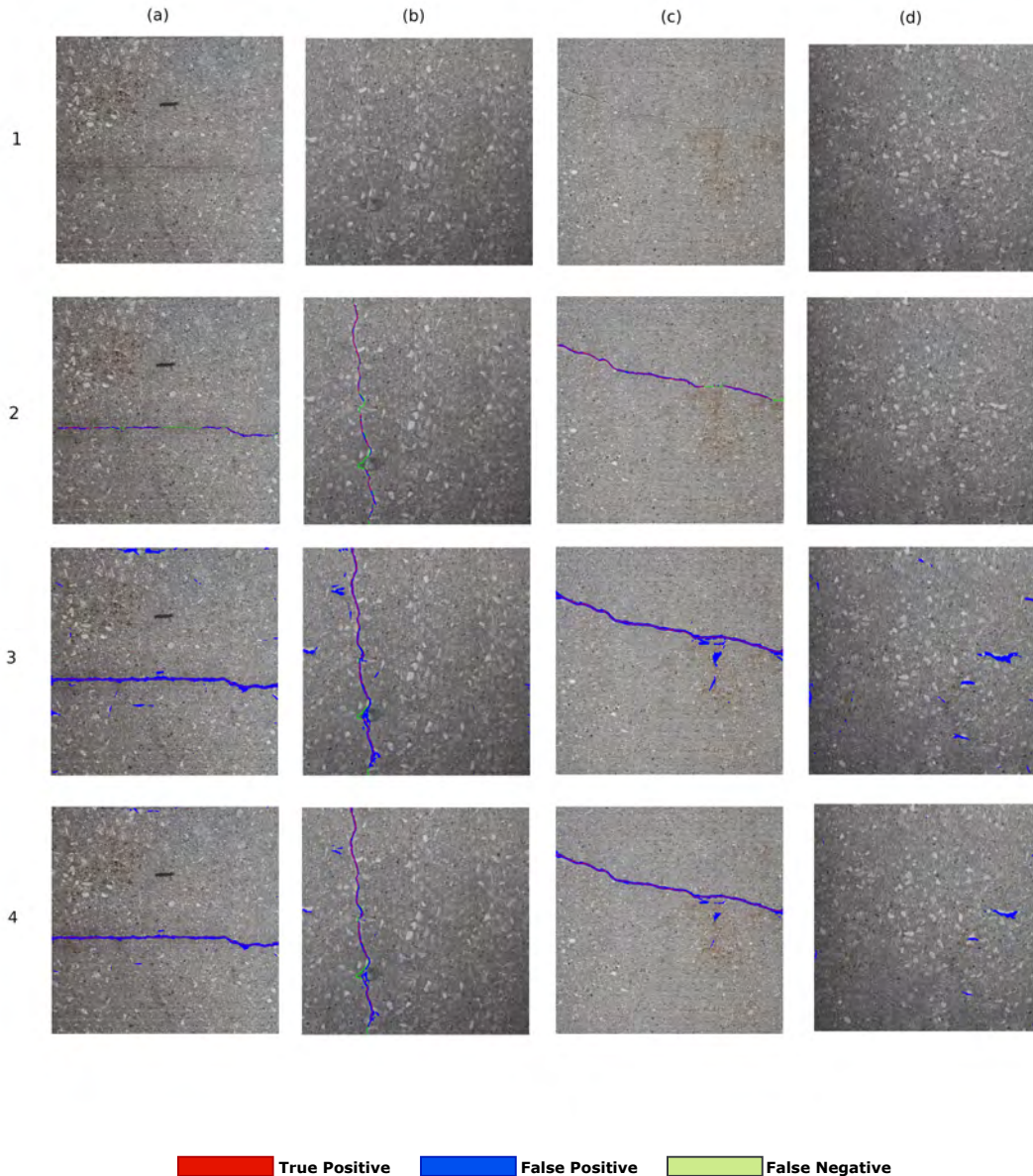


Figure 4.4: Comparison of different thresholds on the performance of ANet-FSM on four sample images. Columns are the tested images: (a) contains clear horizontal crack, (b) contains clear vertical crack, (c) contains a crack at arbitrary orientation, (d) non-cracked concrete image. Row 1- Original image; Row 2- ANet-FSM (hi); Row 3- ANet-FSM (low); and Row 4- ANet-FSM (opt). Results will be seen clearer when zoom-in.

excessive number of feature space (due to maximal network complexity), as well as the vanishing gradients, contribute to this false classification. The SegNet-SO

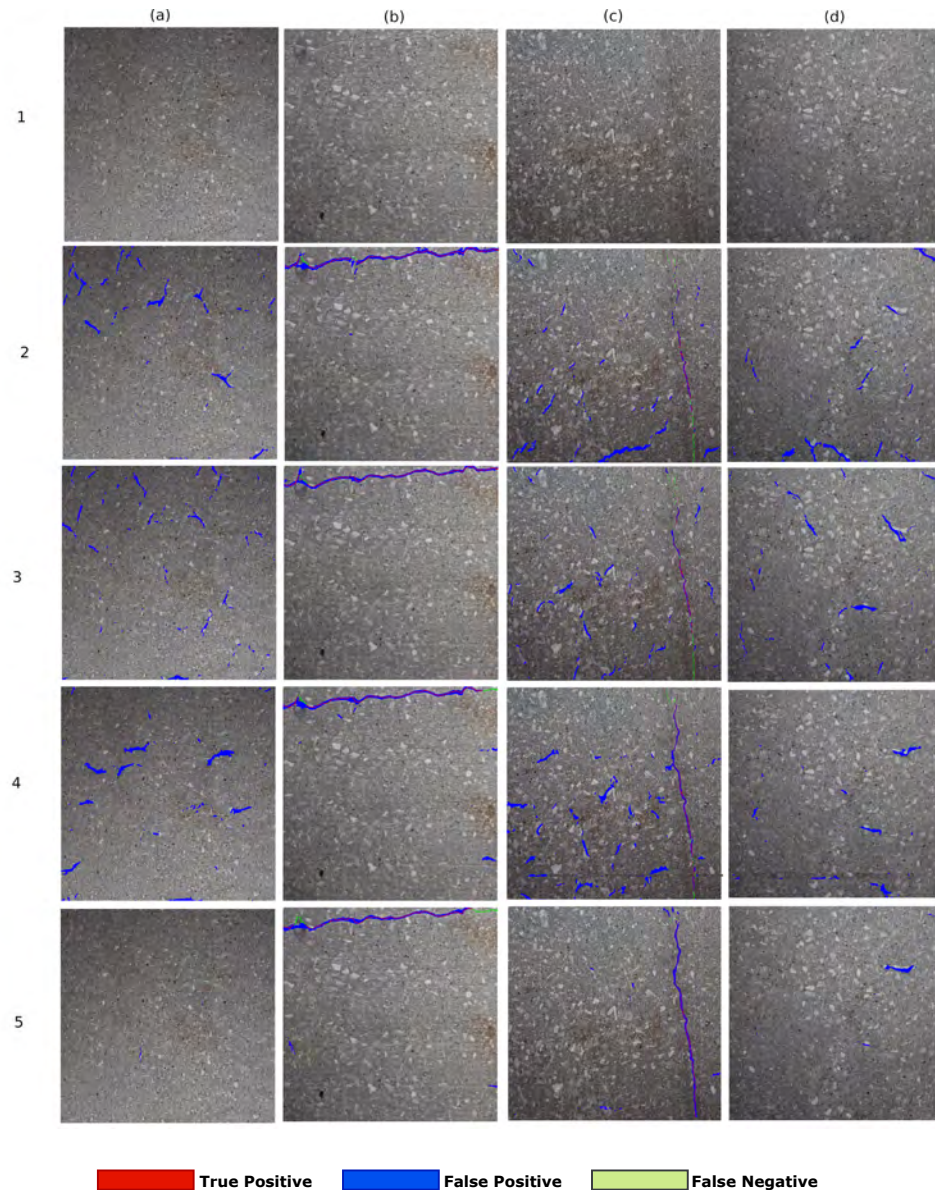


Figure 4.5: Comparison of different encoder-decoder based architectures on four sample images. Columns are the tested images: (a) contains non-cracked concrete image, (b) contains clear horizontal crack on top, (c) contains clear vertical crack, (d) contains non-crack concrete image. Row 1- Original image; Row 2- SegNet [6]; Row 3- SegNet-SO [5]; Row 4- InspectionNet; Row 5- ANet-FSM architecture.

architecture in 3(b) moderately removes the false classification present in image 2(b). The results in 3(a), (c), and (d) represent a considerable amount of falsely classified pixels, specifically in 3(a) where no crack pixels are present originally. On the other

hand, the InspectionNet architecture represented in image 4(a), (b), (c), and (d) shows a performance improvement in comparison to SegNet and SegNet-SO. The results in 4(a) and 4(d) are less affected by the FP rate than SegNet and SegNet-SO. However, the false identification rate increases more than SegNet-SO architecture when a significant number of crack pixels are present as shown in image 3(c). As a result, it can be interpreted that InspectionNet is highly unstable as well as affected by the environmental non-uniformity (lighting and shading). On the other hand, the effect of FPs is significantly low in ANet-FSM architecture in comparison to the results in row-2, row-3, and row-4. It has almost no false identification (blue pixels) in figure 5(a). There exists a small amount of falsely identified pixels in figure 5(d), which is considerably lower than the previous networks. Moreover, this network alleviates this false classification without affecting the correct classification rate (represented as red pixels in 5(b) and (c)), which is the effect of feature silencing. The ANet-FSM architecture not only improves the accuracy of crack identification but also eliminates the effect of false identification significantly with the FSM. Therefore, it can be concluded, ANet-FSM architecture is less affected by the gradient vanishing problem in comparison to the encoder-decoder architectures in [5, 6, 54]. Based on the percentage of correctly identified cracks pixels identified, it can be concluded that ANet-FSM provides performance that is an improvement on the state-of-the-art encoder-decoder network architectures designed for crack detection in the recent past.

4.3.2 Quantitative Comparisons

This section presents the performance of different deep architectures in the literature for concrete crack identification along with ANet-FSM architecture quantitatively. Deep concrete crack detection architectures are categorized as image classification

Table 4.2: Quantitative measures used for evaluating the results of deep network architectures.

| Measure | Definition | Description |
|----------------|------------|---|
| True Positive | TP | Number of accurately identified crack pixels |
| False Positive | FP | Number of falsely classified crack pixels |
| True Negative | TN | Number of accurately classified non-crack pixels |
| False Negative | FN | Number of falsely classified non-crack pixels |
| Accuracy | Acc | $(TP+TN)/(TP+TN+FP+FN)$ |
| Error rate | Err | $(FP+FN)/(TP+TN+FP+FN)$ |
| Specificity | Spc | $TN/(TN+FP)$ |
| Sensitivity | Sens | $TP/(TP+FN)$ |
| Precision | Precision | $TP/(TP+FP)$ |
| Recall | Recall | $TP/(TP+FN)$ |
| F1 score | F1 | $2 \times (\text{Precision} \times \text{Recall})/(\text{Precision}+\text{Recall})$ |

Positive Class = Crack, Negative Class = Non-crack

based architectures (Gibbs [3]) and encoder-decoder based architectures (SegNet [6], InspectionNet [54], SegNet-SO [5]). We evaluated the performance of these networks using the test data-set of 200 images of size 1024×1024 .

Table 4.3: Overall Ranking of each method based on quantitative measures.

| Method | Rank | TPRate | FNRate | TNRate | FPRate | Acc | Err | Spc | Sens | Prec | Rec | RI | F1 | Color Rank |
|--------------------|------|--------|--------|--------|--------|-------|-------|-------|-------|-------|-------|-------|-------|------------|
| Gibbs [2] | 8.0 | 25.2 | 74.8 | 77.0 | 23.0 | 51.1% | 48.9% | 77.0% | 25.2% | 52.3% | 50.7% | 51.1% | 51.5% | 1st |
| SegNet [6] | 6.5 | 71.9 | 28.1 | 98.9 | 1.1 | 85.4% | 14.6% | 98.9% | 71.9% | 98.5% | 77.9% | 85.4% | 87.0% | 2nd |
| SegNet-SO [5] | 4.4 | 73.0 | 27.0 | 99.1 | 0.9 | 86.1% | 14.0% | 99.1% | 73.0% | 98.8% | 78.6% | 86.1% | 87.5% | 3rd |
| InspectionNet [54] | 4.1 | 80.5 | 19.5 | 98.8 | 1.2 | 89.7% | 10.4% | 98.8% | 80.5% | 98.5% | 83.5% | 89.7% | 90.4% | 4th |
| ANet | 4.0 | 75.1 | 24.9 | 99.0 | 1.0 | 87.1% | 13.0% | 99.0% | 75.1% | 98.7% | 79.9% | 87.1% | 88.3% | 5th |
| ANet-FSM (hi) | 4.1 | 72.2 | 27.8 | 99.7 | 0.3 | 86.0% | 14.1% | 99.7% | 72.2% | 99.6% | 78.2% | 86.0% | 87.6% | 6th |
| ANet-FSM (low) | 3.0 | 89.8 | 10.2 | 98.7 | 1.3 | 94.3% | 5.8% | 98.7% | 89.8% | 98.6% | 90.6% | 94.3% | 94.4% | 7th |
| ANet-FSM (opt) | 2.0 | 86.6 | 13.4 | 99.2 | 0.8 | 92.9% | 7.1% | 99.2% | 86.6% | 99.1% | 88.1% | 92.9% | 93.3% | 8th |

In this thesis, the state of the art statistical measures was applied to evaluate the performance of Deep Network architectures. The statistical measures such as True positive (TP) rate, False positive (FP) rate, True negative (TN) rate, False-negative (FN) rate, Error rate, and Accuracy generate biased evaluation results toward non-crack pixels because of crack pixels low appearance. The use of effective statistical measures, such as Specificity, Sensitivity, Precision, Recall, and F-1 measure was employed to distinguish between crack and non-crack pixels within imbalanced datasets. These statistical measures were identified by defining crack pixels as positive

Table 4.4: Ranking of each method based on individual quantitative measures.

| Method | TPRate | FPRate | TNRate | FNRate | Acc | Err | Specificity | Sensitivity | Precision | Recall | F1-Measure |
|--------------------|--------|--------|--------|--------|-----|-----|-------------|-------------|-----------|--------|------------|
| Gibbs [2] | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| SegNet [6] | 7 | 7 | 5 | 5 | 7 | 7 | 5 | 7 | 7 | 7 | 7 |
| SegNet-SO [5] | 5 | 5 | 3 | 3 | 5 | 5 | 3 | 5 | 3 | 5 | 6 |
| InspectionNet [54] | 3 | 3 | 6 | 6 | 3 | 3 | 6 | 3 | 6 | 3 | 3 |
| ANet | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| ANet-FSM (hi) | 6 | 6 | 1 | 1 | 6 | 6 | 1 | 6 | 1 | 6 | 5 |
| ANet-FSM (low) | 1 | 1 | 7 | 7 | 1 | 1 | 7 | 1 | 5 | 1 | 1 |
| ANet-FSM (opt) | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

class and non-crack pixels as negative class. We define the TP rate as the percentage of accurately identified crack pixels, whereas the TN rate is correctly identified non-crack pixels. The FP rates are delineated as the percentage of wrongly identified crack pixels and FNs are the incorrect identification percentage of non-crack pixels. We measure how exactly a method can distinguish between the crack and non-crack pixels with the Precision measure in an imbalanced data-set. We quantify the proportion of crack pixels identified accurately by a network with a recall score. The sensitivity measure evaluates the architecture’s responsiveness toward the aberrant behavior of defected pixels. Specificity measure is used to quantify the behavior of non-crack pixels. The overall performance of a network is evaluated using the F1 score. A summary of the quantitative measures used in this work is shown in table 4.2. Apart from this, we assigned a ranking to the architectures based on both dependent and independent measures. We evaluated the results of the proposed ANet-FSM architecture with three different thresholds. For applications in which high specificity and precision are required, higher threshold values should be utilized—see ANet-FSM (hi) in table 4.4. Lower thresholds would be effective in applications that require higher sensitivity and recall score is needed—see ANet-FSM (low) in table 4.4. In addition to these thresholds, we proposed the use of an optimal threshold, calculated using optimal thresholding algorithms such as the Otsu’s method [69], when no preference is given for performance measure with respect to the positive or the negative classes—see ANet-

FSM (opt) in table 4.4. The overall results of different architecture and their average ranking on all the measures are shown in table 4.3. In addition to this, each network architecture is assigned ranking on individual measures on table 4.4, with 8 being the least accurate. Due to the block-based analysis technique for crack detection, [3] was ranked at 8, where lower rank corresponds to better performance of the algorithm and vice versa. This architecture classifies a sub-image of size 256×256 as crack or non-crack. Since crack pixels occur only a very small portion of an image, an enormous amount of pixels are falsely classified in these blocks. As a result, the false identification rate (both FP rate and FN rate), accuracy as well as the error rate of the Gibbs network is the worst among all the networks. This also represents the inefficiency of image classification methods in concrete crack identification and anomaly detection. As expected, encoder-decoder architectures in table 4.3 and table 4.4, outperform classification networks such as the Gibbs architecture. Although SegNet [6] outperformed all the previous architectures for semantic segmentation in the field of scene parsing, the extremely imbalanced nature of concrete defect dataset greatly drops the performance of this architecture. Additionally, the effect of the gradient vanishing problem (the result of an excessive number of layers) is reflected in evaluation measures such as FP and FN rates. The high false classification rate is also responsible for non-contributing feature maps generated due to the textured nature of the concrete surface (eliminated in ANet-FSM). Furthermore, the SegNet architecture under-performs in all but three measures in table 4.3. For the aforementioned reasons, SegNet architecture is not appropriate for solving the crack identification problem. As a result, SegNet achieves a low overall and individual ranking in all of the evaluation measures in table 4.4 and table 4.3. The gradient vanishing problem affecting the SegNet was alleviated by up-sampling the feature space of each encoder layer in Segnet-SO [5] architecture. This approach achieved more TP rate and less error rate

than SegNet architecture. Although the InspectionNet architecture improves the TP rate significantly, the highest FP rate in table 4.3 demonstrates the effect of gradient vanishing problem. SegNet-SO architecture is more robust to the gradient vanishing problem despite achieving a lower TP rate than InspectionNet. Therefore, it is worth mentioning that, none of the architectures discussed above represent robustness in all the measures.

The ANet-FSM architectures alleviate the drawbacks of SegNet, SegNet-SO, and InspectionNet architectures by eliminating redundant computation. These computationally expensive methods represent a fluctuation in results. For example, InspectionNet obtains outstanding correct classification with the cost of an unacceptable misclassification rate. SegNet-SO degrades the correct classification rate in the course of reducing incorrect classification. To evaluate the stability of the ANet-FSM architecture, we have analyzed its performance without incorporating the FSM (referred to as ANet). The robustness of ANet architecture is reflected by the ranking of 4 in each measure in 4.4. Although the InspectionNet architecture performs better in positive classification (TP), the low negative classification rate (TN) represents the unfeasible nature of this network towards imbalanced data-set. Therefore ANet architecture is substantially stable (the effect of using 7×7 spatial neighborhood in the convolution) despite achieving a lower ranking in some measures. However, the higher false-positive rate of this network than InspectionNet represents that it is affected by the vanishing gradient problem because of a 7×7 kernel size. The association of the FSM model significantly alleviates this problem as well as improves the performance in all measures.

To further investigate the result of feature silencing we performed thresholding operation on the result obtained from the ANet-FSM architecture. We discarded the crack pixels having a lower probability than a specific threshold in this operation.

Table 4.6: Ranked methods based on independent measures.

| Rank | Specificity | Sensitivity | Precision | Recall | F1 |
|------|--------------------|--------------------|--------------------|--------------------|--------------------|
| 1 | ANet-FSM (hi) | ANet-FSM (low) | ANet-FSM (hi) | ANet-FSM (low) | ANet-FSM (low) |
| 2 | ANet-FSM (opt) | ANet-FSM (opt) | ANet-FSM (opt) | ANet-FSM (opt) | ANet-FSM (opt) |
| 3 | SegNet-SO [5] | InspectionNet [54] | SegNet-SO [5] | InspectionNet [54] | InspectionNet [54] |
| 4 | ANet | ANet | ANet | ANet | ANet |
| 5 | SegNet [6] | SegNet-SO [5] | ANet-FSM (low) | SegNet-SO [5] | ANet-FSM (hi) |
| 6 | InspectionNet [54] | ANet-FSM (hi) | InspectionNet [54] | ANet-FSM (hi) | SegNet-SO [5] |
| 7 | ANet-FSM (low) | SegNet [6] | SegNet [6] | SegNet [6] | SegNet [6] |
| 8 | Gibbs [2] | Gibbs [2] | Gibbs [2] | Gibbs [2] | Gibbs [2] |

Gibbs method [3] and SegNet [6] architecture have the lowest precision and recall score in table 4.4 and 4.3. SegNet-SO architecture has a higher precision rate than InspectionNet architecture. However, the recall score represents a reverse relationship between SegNet-SO and InspectionNet. This tension between precision and recall is a well-known phenomenon within classification problems suffering from class-imbalance issues. Moreover, specificity and sensitivity represent similar relationships as precision and recall due to excessive class imbalance present in concrete crack data-sets. If a network is highly specific, its sensitivity reduces (SegNet-SO) whereas a high sensitivity rate reduces the specificity of a network(InspectionNet). As a result, the F1 score is widely used to combine the effects of these measures for any machine learning architecture. The F1 score of SegNet-SO and InspectionNet represents that the former is better in terms of overall performance.

On the other hand, the ANet architecture maintains a stable precision, recall, specificity, and sensitivity scores (all are assigned the same rank in table 4.4). Consequently, its F1 score is better than SegNet-SO and less than InspectionNet because of lower sensitivity. However, the ANet-FSM architecture with a low threshold achieves exceptional recall scores with relatively low specificity, resulting in the highest F1

score of all the methods. If extreme thresholding is applied, ANet-FSM architecture obtains the highest precision and specificity score with moderately low recall and sensitivity score among all the architectures. The optimal thresholding operation achieves higher precision, recall, specificity, sensitivity, and F1 scores than all the networks in table 4.3 and table 4.4. The same ranking in all of the measures in table 4.4 also demonstrates the stability of the network. This architecture obtains the highest F1-score among all the computationally expensive networks (SegNet, InspectionNet, SegNet-SO). Therefore, it can be concluded that this network is appropriate for use in applications with highly class-imbalanced data.

We have also represented a ranked position of different deep network architectures for dependent and independent measures in table 4.5 and table 4.6, respectively. According to our earlier propositions of dependent measures, it is evident from table 4.5, the ANet-FSM architecture outperforms the existing encoder-decoder architecture for defect identification despite the bias towards the non-crack class. The independent measures also represent that the ANet-FSM architecture outperforms the other existing networks in table 4.6. Moreover, ANet-FSM (hi) is suitable for applications requiring extremely specific and precise results. ANet-FSM (low) is appropriate for highly sensitive applications for concrete crack detection. The robustness of ANet-FSM (opt) toward specificity, sensitivity, precision, and recall makes this suitable for applications that require stable results.

We have also compared the performance of ANet-FSM architecture with state-of-the-art crack detection architectures such as DeepCrack [4] and SDDNet [55]. The results of deep crack and SDDNet were extracted from the experiments reported in [4]. The ANet-FSM architecture was trained and tested using the data-set [4] used for the experiment in [55]. For comparison metrics, we have taken into consideration the precision, recall, F1 score, and the processing time. The results are shown in table 4.7.

Table 4.7: Comparison ANet-FSM architecture with crack detection architectures. The ANet-FSM architecture was trained and tested on the data-set prepared by DeepCrack [4].

| Method | Prec | Rec | F1 | Processing time |
|---------------|--------|--------|--------|-----------------|
| DeepCrack [4] | 86.10% | 86.90% | 86.50% | 109 ms |
| SDDNet [55] | 87.10% | 87.00% | 87.00% | 13.54ms |
| ANet-FSM | 98.2% | 78.2% | 87.1% | 1.14ms |

SDDNet architecture achieves the highest F1 score among all the methods. However, the ANet-FSM architecture achieves the highest precision rate and an F1 score close to SDDNet. This phenomenon represents the effect of gradient vanishing in lowest in ANet-FSM among all the methods. Moreover, the processing time of ANet-FSM is 1.14 ms per image, whereas SDDNet has 13.04 ms per image. The processing speed of ANet-FSM architecture is thirteen-time smaller than the SDDNet architecture. Considering this significant fast processing time of ANet-FSM architecture, the lower F1 score is reasonable. Apart from this, this processing time also depicts ANet-FSM architecture is nominally affected by gradient stability problem.

4.4 Network Complexity Analysis

One of the solution for tackling the gradient vanishing problem of very deep networks is the reduction of computational complexity. Since the convolutional layers are responsible for salient feature attribute extraction in CNN architecture [40], most of the computations are performed by this layer. As a result, the complexity of the network was measured based on the computations associated with the convolutional layers. The network complexity is defined as C in equation 4.1

$$C = \sum_{i=0}^{E+D} (N_{C_i} \times K_s) \times N_{k_i} \quad (4.1)$$

where N_C is the number of convolution layers, N_k is the number of kernels in each layer, K_s is a $m \times n$ dimensional kernel (m is height and n is width), E is the total number of encoders and D is the total number of decoders. Since the network architectures discussed in this work use five encoders and five decoders we set the value of $E = 5$ and $D = 5$.

Table 4.8: Comparison of network complexity of different encoder decoder based architecture. C : network complexity, N_C : number of convolutional layers N_k : number of kernels in each convolutional layer, K_s : a $m \times n$ dimension kernel

| Method | Nc | Nk | Ks | C |
|---------------------------|--------|----|-----|-------|
| VGG-16 [47] | {†} | ⊕ | 3x3 | 38016 |
| SegNet [6] | {†, ‡} | ∗ | 3x3 | 72576 |
| InspectionNet [54] | ⋈ | ∨ | 3x3 | 34560 |
| ANet-FSM | ⊕ | △ | 7x7 | 12152 |

No. of Convs. per Encoder (Nc) :

† = {2,2,3,3,3}, ‡ = {3,3,3,2,2},

⋈ = {2,2,2,2,2}, ⊕ = {1,1,1,1,1}

No. of Filter Sets per Encoder (Nk) :

⊕ = {64, 128, 256, 512, 512}

∗ = {64, 128, 256, 512, 512, 512, 256, 128, 64}

∨ = {64, 128, 256, 512, 512, 256, 128, 64}

◇ = {8, 16, 32, 64, 64, 64, 32, 16, 8}

△ = {4, 8, 16, 32, 64, 64, 32, 16, 8, 4}

To analyze the network complexity we take into consideration three encoder-decoder architectures in literature, i.e., SegNet [6] and InspectionNet [54]. Since the encoder and decoder networks in SegNet architecture follow the topology of VGG-16 architecture, we computed the complexity of this network. The number of computa-

tions required for the ANet-FSM architecture is compared with the aforementioned network architectures in table 4.8.

The number of total computations performed by VGG-16 architecture is 38016. Both the encoder and decoder network in SegNet follows VGG-16 architecture topology. Consequently, the number of convolution layers, kernels, and final computations increase by two times of the original VGG-16 architecture. The InspectionNet architecture performs 3806 fewer computations than SegNet. On the other hand, the ANet architecture performs 12152 computations. Despite using a larger convolution kernel this architecture performs 22408 fewer computations than InspectionNet. Usage of a single convolution layer in each encoder along with less filter numbers, reduced the computational complexity of ANet-FSM architecture. This aspect of the network helps eliminating the effect of gradient vanishing problem of deep networks. Furthermore, the precision of the network increases significantly due to this fact. Therefore, ANet-FSM architecture is the most in-expensive network in terms of computation in comparison to the networks represented in table 4.8.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

A deep convolutional neural network for concrete crack classification is presented in this work. A framework for eliminating the effect of gradient vanishing problem for class imbalanced data-set is presented in this work. Silencing unnecessary feature space enhances the precision of our framework as well as reduce the effect of gradient vanishing problem. Moreover, the sensitivity of the architecture to environmental non-uniformity is reduced by the FSM module. As a result, our architecture is more precise than the crack detection methods present in the literature. The experimental results in this study also represent that an enormous amount of unnecessary computations is performed in deep architectures. Elimination of these computations enhances the speed and processing of deep networks remarkably, which is important for the real-time deployment of these architectures. The FSM module selects a feature based on some threshold in the proposed framework.

5.2 Future Work

Although the proposed framework achieves the highest precision in crack detection, the silencing policy is handcrafted. As a result, the framework is highly sensitive to a certain type of data-set. This sensitivity can be alleviated by designing an optimization framework, enabling the network to choose policies based on data-set criteria. Another future direction of this work would be exploring the area of other concrete distress identification such as spalling detection using an optimized feature silencing module. The proposed framework for crack detection in this work is appropriate for identifying a crack location. However, civil infrastructure inspection requires the quantification of defects also. Quantifying cracks involve identifying the length and depth of the crack. The proposed framework is appropriate for identifying crack length. Quantifying the depth measurement of crack requires depth data. Therefore, another future direction of this work is identifying the severity of any defect based on depth data.

Bibliography

- [1] U. H. Billah, H. M. La, A. Tavakkoli, and N. Gucunski, “Classification of concrete crack using deep residual network,” in *9th International Conference on Structural Health Monitoring of Intelligent Infrastructure (SHMII-9)*, Aug. 2019, pp. 1–6.
- [2] Y.-J. Cha, W. Choi, and O. Büyüköztürk, “Deep learning-based crack damage detection using convolutional neural networks,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, pp. 361–378, 2017.
- [3] S. Gibb, H. M. La, T. Le, L. Nguyen, R. Schmid, and H. Pham, “Nondestructive evaluation sensor fusion with autonomous robotic system for civil infrastructure inspection,” *Journal of Field Robotics*, vol. 35, no. 6, pp. 988–1004, 2018.
- [4] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang, “Deepcrack: Learning hierarchical convolutional features for crack detection,” *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1498–1512, 2018.
- [5] U. H. Billah, A. Tavakkoli, and H. M. La, “Concrete crack pixel classification using an encoder decoder based deep learning architecture,” in *International Symposium on Visual Computing*. Springer, 2019, pp. 593–604.

- [6] V. Badrinarayanan, A. Handa, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling,” *arXiv preprint arXiv:1505.07293*, 2015.
- [7] H. M. La, N. Gucunski, S. H. Kee, and L. Nguyen, “Visual and acoustic data analysis for the bridge deck inspection robotic system,” *The 31st Inter. Sympo. on Automation and Robotics in Construction and Mining (ISARC)*, 2014.
- [8] N. Gucunski, S. H. Kee, H. M. La, B. Basily, and A. Maher, “Delamination and concrete quality assessment of concrete bridge decks using a fully autonomous rabbit platform,” *International Journal of Structural Monitoring and Maintenance*, vol. 2, no. 1, pp. 19–34, April 2015.
- [9] S. Gibb and H. M. La, “Automated rebar detection for ground-penetrating radar,” in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koricin, F. Porikli, S. Skaff, A. Entezari, J. Min, D. Iwai, A. Sadagic, C. Scheidegger, and T. Isenberg, Eds. Cham: Springer International Publishing, 2016, pp. 815–824.
- [10] H. M. La, N. Gucunski, S. H. Kee, and L. V. Nguyen, “Data analysis and visualization for the bridge deck inspection and evaluation robotic system,” *Springer Journal of Visualization in Engineering*, vol. 3, no. 6, pp. 1–16, Feb 2015.
- [11] H. M. La, R. S. Lim, B. B. Basily, N. Gucunski, J. Yi, A. Maher, F. A. Romero, and H. Parvardeh, “Autonomous robotic system for high-efficiency non-destructive bridge deck inspection and evaluation,” *IEEE Inter. Conf. on Auto. Sci. and Eng. (CASE)*, pp. 1065–1070, 2013.
- [12] H. M. La, R. S. Lim, B. B. Basily, N. Gucunski, J. Yi, A. Maher, F. A. Romero, and H. Parvardeh, “Mechatronic systems design for an autonomous robotic sys-

- tem for high-efficiency bridge deck inspection and evaluation,” *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1655–1664, Dec 2013.
- [13] H. M. La, N. Gucunski, Seong-Hoon Kee, J. Yi, T. Senlet, and Luan Nguyen, “Autonomous robotic system for bridge deck data collection and analysis,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1950–1955.
- [14] H. M. La, N. Gucunski, K. Dana, and S.-H. Kee, “Development of an autonomous bridge deck inspection robotic system,” *Journal of Field Robotics*, vol. 34, no. 8, pp. 1489–1504, 2017. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21725>
- [15] N. Sy, M. Avila, S. Begot, and J.-C. Bardet, “Detection of defects in road surface by a vision system,” in *Electrotechnical Conference, 2008. MELECON 2008. The 14th IEEE Mediterranean*. IEEE, 2008, pp. 847–851.
- [16] Q. Li and X. Liu, “Novel approach to pavement image segmentation based on neighboring difference histogram method,” in *Image and Signal Processing, 2008. CISP’08. Congress on*, vol. 2. IEEE, 2008, pp. 792–796.
- [17] H. Oliveira and P. L. Correia, “Automatic road crack segmentation using entropy and image dynamic thresholding,” in *Signal Processing Conference, 2009 17th European*. IEEE, 2009, pp. 622–626.
- [18] T. H. Dinh, Q. P. Ha, and H. M. La, “Computer vision-based method for concrete crack detection,” in *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Nov 2016, pp. 1–6.
- [19] A. Tavakkoli, M. Nicolescu, and G. Bebis, “Automatic robust background modeling using multivariate non-parametric kernel density estimation for visual surveil-

- lance,” in *International Symposium on Visual Computing*. Springer, 2005, pp. 363–370.
- [20] Y. Sun, E. Salari, and E. Chou, “Automated pavement distress detection using advanced image processing techniques,” in *Electro/Information Technology, 2009. Eit’09. IEEE International Conference on*. IEEE, 2009, pp. 373–377.
- [21] A. Landstrom and M. J. Thurley, “Morphology-based crack detection for steel slabs,” *IEEE Journal of selected topics in signal processing*, vol. 6, no. 7, pp. 866–875, 2012.
- [22] X. Bai, F. Zhou, and B. Xue, “Multiple linear feature detection based on multiple-structuring-element center-surround top-hat transform,” *Applied optics*, vol. 51, no. 21, pp. 5201–5211, 2012.
- [23] H. Elbehiery, A. Hefnawy, and M. Elewa, “Surface defects detection for ceramic tiles using image processing and morphological techniques,” 2005.
- [24] N. Tanaka and K. Uematsu, “A crack detection method in road surface images using morphology.” *MVA*, vol. 98, pp. 17–19, 1998.
- [25] Y. Maode, B. Shaobo, X. Kun, and H. Yuyao, “Pavement crack detection and analysis for high-grade highway,” in *Electronic Measurement and Instruments, 2007. ICEMI’07. 8th International Conference on*. IEEE, 2007, pp. 4–548.
- [26] H. Zhao, G. Qin, and X. Wang, “Improvement of canny algorithm based on pavement edge detection,” in *2010 3rd International Congress on Image and Signal Processing*, vol. 2. IEEE, 2010, pp. 964–967.
- [27] R. S. Lim, H. M. La, Z. Shan, and W. Sheng, “Developing a crack inspection robot for bridge maintenance,” in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 6288–6293.

- [28] P. Prasanna, K. Dana, N. Gucunski, and B. Basily, "Computer-vision based crack detection and analysis," in *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2012*, vol. 8345. International Society for Optics and Photonics, 2012, p. 834542.
- [29] R. S. Lim, H. M. La, and W. Sheng, "A robotic crack inspection and mapping system for bridge deck maintenance," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 367–378, 2014.
- [30] H. Kim, E. Ahn, S. Cho, M. Shin, and S.-H. Sim, "Comparative analysis of image binarization methods for crack identification in concrete structures," *Cement and Concrete Research*, vol. 99, pp. 53–61, 2017.
- [31] A. Tavakkoli, M. Nicolescu, and G. Bebis, "An adaptive recursive learning technique for robust foreground object detection," in *the International Workshop on Statistical Methods in Multi-image and Video Processing (in conjunction with ECCV06)*, 2006, pp. 1–6.
- [32] A. Tavakkoli, A. Ambardekar, M. Nicolescu, and S. Louis, "A genetic approach to training support vector data descriptors for background modeling in video data," in *International Symposium on Visual Computing*. Springer, 2007, pp. 318–327.
- [33] M. Gavilán, D. Balcones, O. Marcos, D. F. Llorca, M. A. Sotelo, I. Parra, M. Ocaña, P. Aliseda, P. Yarza, and A. Amírola, "Adaptive road crack detection system by pavement classification," *Sensors*, vol. 11, no. 10, pp. 9628–9657, 2011.
- [34] P. Prasanna, K. J. Dana, N. Gucunski, B. B. Basily, H. M. La, R. S. Lim, and H. Parvardeh, "Automated crack detection on concrete bridges," *IEEE Trans-*

- actions on Automation Science and Engineering*, vol. 13, no. 2, pp. 591–599, 2016.
- [35] J. Bray, B. Verma, X. Li, and W. He, “A neural network based technique for automatic classification of road cracks,” in *Neural Networks, 2006. IJCNN’06. International Joint Conference on*. IEEE, 2006, pp. 907–912.
- [36] L. Wu, S. Mokhtari, A. Nazef, B. Nam, and H.-B. Yun, “Improvement of crack-detection accuracy using a novel crack defragmentation technique in image-based road assessment,” *Journal of Computing in Civil Engineering*, vol. 30, no. 1, p. 04014118, 2014.
- [37] M. O’Byrne, B. Ghosh, F. Schoefs, and V. Pakrashi, “Regionally enhanced multi-phase segmentation technique for damaged surfaces,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 29, no. 9, pp. 644–658, 2014.
- [38] A. Banharsakun, “Hybrid abc-ann for pavement surface distress detection and classification,” *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 2, pp. 699–710, 2017.
- [39] H. Moon, J. Kim *et al.*, “Intelligent crack detecting algorithm on the concrete crack image using neural network,” *Proceedings of the 28th ISARC*, pp. 1461–1467, 2011.
- [40] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.
- [41] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

- [42] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 4278–4284.
- [43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [44] B. Kim and S. Cho, “Automated vision-based detection of cracks on concrete surfaces using a deep learning technique,” *Sensors*, vol. 18, no. 10, p. 3452, 2018.
- [45] X. Zhao, S. Li, H. Su, L. Zhou, and K. J. Loh, “Image-based comprehensive maintenance and inspection method for bridges using deep learning,” in *ASME 2018 Conference on Smart Materials, Adaptive Structures and Intelligent Systems*. American Society of Mechanical Engineers Digital Collection, 2018.
- [46] K. Jang, N. Kim, and Y.-K. An, “Deep learning-based autonomous concrete crack evaluation through hybrid image scanning,” *Structural Health Monitoring*, vol. 18, no. 5-6, pp. 1722–1737, 2019.
- [47] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [48] S. Gibb, H. M. La, and S. Louis, “A genetic algorithm for convolutional network structure optimization for concrete crack detection,” in *2018 IEEE Congress on Evolutionary Computation (CEC)*, July 2018, pp. 1–8.
- [49] C. V. Dung *et al.*, “Autonomous concrete crack detection using deep fully convolutional neural network,” *Automation in Construction*, vol. 99, pp. 52–58, 2019.

- [50] Y. Li, W. Zhao, X. Zhang, and Q. Zhou, “A two-stage crack detection method for concrete bridges using convolutional neural networks,” *IEICE TRANSACTIONS on Information and Systems*, vol. 101, no. 12, pp. 3249–3252, 2018.
- [51] J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks,” in *Advances in neural information processing systems*, 2016, pp. 379–387.
- [52] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.
- [53] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
- [54] L. Yang, B. Li, W. Li, B. Jiang, and J. Xiao, “Semantic metric 3d reconstruction for concrete inspection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1543–1551.
- [55] W. Choi and Y.-J. Cha, “Sddnet: Real-time crack segmentation,” *IEEE Transactions on Industrial Electronics*, 2019.
- [56] S. Bang, S. Park, H. Kim, and H. Kim, “Encoder–decoder network for pixel-level road crack detection in black-box images,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 8, pp. 713–727, 2019.
- [57] J. Zhang, C. Lu, J. Wang, L. Wang, and X.-G. Yue, “Concrete cracks detection based on fcn with dilated convolution,” *Applied Sciences*, vol. 9, no. 13, p. 2686, 2019.

- [58] H. Li, D. Song, Y. Liu, and B. Li, "Automatic pavement crack detection by multi-scale image fusion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2025–2036, 2018.
- [59] C. V. Dung, H. Sekiya, S. Hirano, T. Okatani, and C. Miki, "A vision-based method for crack detection in gusset plate welded joints of steel bridges using deep convolutional neural networks," *Automation in Construction*, vol. 102, pp. 217–229, 2019.
- [60] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *2016 IEEE international conference on image processing (ICIP)*. IEEE, 2016, pp. 3708–3712.
- [61] W. Liu, Y. Huang, Y. Li, and Q. Chen, "Fpcnet: Fast pavement crack detection network based on encoder-decoder architecture," *arXiv preprint arXiv:1907.02248*, 2019.
- [62] H. Kim, E. Ahn, M. Shin, and S.-H. Sim, "Crack and noncrack classification from concrete surface images using machine learning," *Structural Health Monitoring*, vol. 18, no. 3, pp. 725–738, 2019.
- [63] Q. Mei and M. Gül, "Multi-level feature fusion in densely connected deep-learning architecture and depth-first search for crack segmentation on images collected with smartphones," *Structural Health Monitoring*, p. 1475921719896813, 2020.
- [64] M. Azimi, A. D. Eslamlou, and G. Pekcan, "Data-driven structural health monitoring and damage detection through deep learning: State-of-the-art review," *Sensors*, vol. 20, no. 10, p. 2778, 2020.
- [65] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1395–1403.

- [66] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang, “Deepcrack: Learning hierarchical convolutional features for crack detection,” *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1498–1512, 2018.
- [67] S. Gibb, T. Le, H. M. La, R. Schmid, and T. Berendsen, “A multi-functional inspection robot for civil infrastructure evaluation and maintenance,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 2672–2677.
- [68] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen, “Automatic road crack detection using random structured forests,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3434–3445, 2016.
- [69] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.