STRUCTURAL CRACK DETECTION USING DEEP CONVOLUTIONAL NEURAL NETWORK

RAZA ALI

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2022

STRUCTURAL CRACK DETECTION USING DEEP CONVOLUTIONAL NEURAL NETWORK

RAZA ALI

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2022

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **Raza Ali**

Matric No: **KVA180046 / 17198060**

Name of Degree: **Doctor of Philosophy (Ph.D.)**

Title of Thesis: **STRUCTURAL CRACK DETECTION USING DEEP CONVOLUTIONAL NEURAL NETWORK**

Field of Study: Signal processing, Computer vision, and Deep learning

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

STRUCTURAL CRACK DETECTION USING DEEP CONVOLUTIONAL NEURAL NETWORK

ABSTRACT

Convolutional Neural Networks (CNN) have immense potential to solve a broad range of computer vision (CV) problems. It has achieved encouraging results in numerous applications in engineering, medical, and other research fields. Thanks to the advancement in hardware, data collection procedures, and efficient algorithms. These innovations have changed the way how specific problems are solved as compared to conventional methods. In this work, CNN is implemented for civil structural crack detection. Cracks are significant indicators for the evaluation of the structural health and monitoring process. However, manual crack detection is a time-consuming and challenging task due to large areas, complex structures, and safety risks. Deep learning (DL) has emerged as an effective technique to automate the crack detection and identification process. For balanced data, existing DL models attempt to segment both crack pixels and non-crack pixels equally. However, due to the highly imbalanced ratio between crack pixels and non-crack pixels, the pixel-wise loss is dominantly guided by the non-crack region and has relatively little influence from the crack region. This leads to the low segmentation accuracy for crack pixels. To address the imbalance problem, this work proposes a local weighting factor with a difference transform map to remove the network biasness and accurately predict the sensitive pixels. Further, a deep fully CNN called crack segmentation network (CSN) is implemented for crack pixel segmentation. The CSN is an encoder-decoder architecture with four convolutional blocks in each section. Each convolutional block has residual connections with a different number of filters in each convolutional operation that segments the crack pixels and non-crack pixels with unbiased probabilities. Furthermore, the crack indicators are assessed through the implementation of a pixel connection technique that measures the crack

characteristics (length, width, and area) and determines the crack orientation (vertical, horizontal, and diagonal). For performance evaluation, a new Multi Structure Crack Image (MSCI) dataset is built to train the proposed method which achieved 98.60% crack pixel accuracy, 98.35% non-crack pixel accuracy, and 98.48% average accuracy, respectively. In addition, the training time for 10 epochs has dramatically decreased and the experimental results show that the proposed CSN architecture has better crack pixel segmentation accuracy than FCN, U-Net, SegNet, and DeepLabv3+ architectures. Similarly, the proposed local weighting factor and difference transform map (LWF-DTM) has significantly reduced the wrong predictions, minimized the effect of an imbalanced pixel ratio, and outperformed the Cross-Entropy, Weighted Cross-Entropy, Dice, Tversky, and Focal loss function.

Keywords: Deep learning, crack detection, Imbalanced dataset, Loss functions, Residual blocks, Pixel local weights.

PENGESANAN REHAK STRUKTUR MENGGUNAKAN RANGKAIAN SARAF SAMBUNGAN DALAM

ABSTRAK

Rangkaian Neural Konvolusi (CNN) mempunyai potensi besar untuk menyelesaikan pelbagai masalah penglihatan komputer. Ia telah mencapai keputusan yang menggalakkan dalam pelbagai aplikasi bidang kejuruteraan, perubatan dan penyelidikan lain disebabkan oleh kemajuan dalam perkakasan, prosedur pengumpulan data dan algoritma yang cekap. Inovasi ini telah mengubah cara bagaimana masalah khusus diselesaikan berbanding kaedah konvensional. Dalam kerja ini, CNN dilaksanakan untuk pengesanan retak struktur awam. Retak adalah petunjuk penting untuk penilaian kesihatan struktur dan proses pemantauan. Walau bagaimanapun, pengesanan retak manual adalah tugas yang memakan masa dan mencabar kerana kawasan yang luas, struktur yang kompleks dan risiko keselamatan. Pembelajaran mendalam (DL) telah muncul sebagai teknik yang berguna untuk mengautomasikan proses pengesanan dan pengecaman retak. Untuk data seimbang, model DL sedia ada cuba membahagikan kedua-dua piksel retak dan piksel bukan retak secara sama rata. Walau bagaimanapun, disebabkan nisbah yang sangat tidak seimbang antara piksel retak dan piksel bukan retak, kehilangan dari segi piksel secara dominan dipandu oleh rantau bukan retak dan mempunyai pengaruh yang agak kecil dari rantau retak. Ini membawa kepada ketepatan pembahagian yang rendah untuk piksel retak. Untuk menangani masalah ketidakseimbangan, kerja ini mencadangkan faktor pemberat setempat dengan peta sensitiviti untuk mengalih keluar bias rangkaian dan meramalkan piksel sensitif dengan tepat. Selanjutnya, CNN sepenuhnya yang dalam dipanggil rangkaian segmentasi retak (CSN) dilaksanakan untuk segmentasi piksel retak. CSN ialah seni bina penyahkod pengekod dengan empat blok konvolusi dalam setiap bahagian. Setiap blok konvolusi mempunyai sambungan baki dengan bilangan penapis yang berbeza dalam setiap operasi

konvolusi yang membahagikan piksel retak dan piksel bukan retak dengan kebarangkalian tidak berat sebelah. Tambahan pula, penunjuk retak dinilai melalui pelaksanaan teknik sambungan piksel yang mengukur ciri retak (panjang, lebar, dan luas) dan menentukan orientasi retak (menegak, mendatar, dan pepenjuru). Untuk penilaian prestasi, set data Multi Structure Crack Image (MSCI) baharu dibina untuk melatih kaedah yang dicadangkan dimana ia mencapai 98.60% ketepatan piksel retak, 98.35% ketepatan piksel bukan retak dan 98.48% ketepatan purata. Di samping itu, masa latihan untuk 10 zaman telah berkurangan secara mendadak dan keputusan percubaan menunjukkan bahawa seni bina CSN yang dicadangkan mempunyai ketepatan pembahagian piksel retak yang lebih baik daripada seni bina U-Net dan SegNet. Begitu juga, faktor pemberat tempatan dan peta transformasi perbezaan (LWF-DTM) yang dicadangkan telah mengurangkan ramalan yang salah dengan ketara, meminimumkan kesan nisbah piksel yang tidak seimbang, dan mengatasi fungsi kehilangan Cross-Entropi, Cross-Entropi Berwajaran, Dadu, Tversky dan Focal.

Kata kunci: Pembelajaran mendalam, pengesanan retak, set data yang tidak seimbang, fungsi kehilangan, blok sisa, pemberat tempatan piksel.

ACKNOWLEDGEMENTS

ALHAMDULILLAH, all praises to Almighty سُبْحَانَهُ وَتَعَالَى, for making this Ph.D. journey learnable, easy, and successful for me.

I would like to thank my supervisor Ir. Dr. Chuah Joon Huang for his support, guidance, and encouragement from the beginning to the end of this journey. Without his continuous support in every stage of this research work, I may never have accomplished my thesis. I am also deeply grateful to my co-supervisors.

I would also like to express my greatest thanks to all my family members, especially Mr. Tahir Mahmood (my father), Mrs. Sajjida Tahir (my mother), my brother, and my sisters for their prayers, support, and encouragement throughout my Ph.D. journey. Most importantly, my heartful thanks to my life partner, Mrs. Fahmeena Raza, and my junior Muhammad Shees for being my companion during this period of life.

I would like to express appreciation to all my friends, colleagues, and those who either directly or indirectly gave cooperation, motivation, and support along my journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

AE	:	Auto Encoder
AI	:	Artificial Intelligence
CBFR	:	Component-Based Face Recognition
CCR	:	Crack Candidate Region
CCTV	:	Closed Circuit Television
CDN	:	Crack Delineation Network
CMOS	:	Complementary Metal Oxide Semiconductor
CNN	:	Convolutional Neural Network
CPU	:	Central Processing Unit
CRF	:	Conditional Random Fields
CV	:	Computer Vision
DBN	:	Deep Belief Network
DCNN	:	Deep Convolutional Neural Network
DL	:	Deep Learning
DNN	:	Deep Neural Networks
DSN	:	Deeply Supervised Networks
ERT	:	Electrical Resistance Tomography
FCN	:	Fully Connected Network
GAN	:	Generative Adversarial Network
GF	:	Guided Filtering
GPU	:	Graphics Processing Unit
GRU	:	Gated Recurrent Units
IoU	:	Intersection over the Union
ILSVRC	:	ImageNet Large Scale Visual Recognition Challenge
LReLU	:	Leaky Rectified Linear Unit
LSTM	:	Long Short-Term Memory

ML	:	Machine Learning
MPL	:	Multi-Layered Perceptron
NiN	:	Network in Network
NN	:	Neural Networks
PCA	:	Principal Component Analysis
RBM	:	Restricted Boltzmann Machine
RCNN	:	Region Convolutional Neural Network
ReLU	:	Rectified Linear Unit
ResNet	:	Residual Network
RGB	:	Red, Green, and Blue
RNN	:	Recursive Neural Networks
SGD	:	Stochastic Gradient Descent
SHM	:	Structural Health Monitoring
SSD	:	Single Shot Multi-Box Detector
SURF	:	Speeded-Up Robust Features
SVM	:	Support Vector Machine
VGG	:	Visual Geometry Group
YOLO	:	You Only Look Once

LIST OF APPENDICES

CHAPTER 1: INTRODUCTION

1.1 Overview

Structural Health Monitoring (SHM) refers to a regular check-up mechanism to monitor the condition and characteristics of civil structures. SHM system is developed to monitor the state of health of a civil structure compared to normal conditions. Its primary purpose is to identify the changes in the civil structure, devise a maintenance plan and take appropriate action against the abnormality in the structure (Taheri, 2019). However, there are many ways (Guo et al., 2017) to monitor structural health through multiple structure health indices. On the other hand, damage detection can be interpreted as a structural modification that changes physical properties and weakens the structure. A crack is generally defined as a defect that may cause serious damage and consequences to the structure. For the safety and maintenance of roads, subways, bridges, buildings, dams, tunnels, monuments, etc., cracks are one of the factors that determine the structural condition. It is considered an initial indication that exhibits the degradation of civil structures. Cracks can appear on any of the civil structures and can occur due to various reasons such as low material quality, improper maintenance, and atmospheric effect. Early detection can help prevent bigger disasters and ensure the safety of the civil structure.

Detection using manual inspection requires skilled (experience and knowledge) labor with appropriate tools to regularly inspect and identify the cracks by following the safety precautions, which makes this activity very costly and time-consuming. For large and high infrastructures, manual detection is a safety risk and a lengthy process. To automate the crack segmentation process, various methods such as image processing techniques (IPT), machine learning (ML), and deep learning (DL) techniques are deployed. IPTs involve image pre-processing, segmentation, extraction of features from the image, and crack recognition to detect the cracks. IPTs use edges information (Abdel-Qader et al.,

2003), morphological operations (Yamaguchi et al., 2008), digital image correlation (Gehri et al., 2020), graph-based method (Koch et al., 2014), and pattern matching (Adhikari et al., 2014a) for crack detection. Crack detection through image processing techniques encounters several environmental challenges such as shadow, dust, stain noise, multicolor spots, uneven light illumination, and multiple background scenes. Change in the dataset also creates resistance to detecting cracks from the images. The problems with IPTs based algorithms are; the image orientation effect, lack of in-depth information, the distance between source and camera, noise in the image, regions with similar crack features, human involvement, and change in dataset degrades the algorithm performance.

Meanwhile ML uses manual features and targets those areas in the image that contains cracks. ML uses different approaches to classify crack features, and the most common methods are support vector machine (SVM) (Varadharajan et al., 2014), Markov's based method (Delagnes & Barba, 2002), image binarization (Ahmadi et al., 2018a) and random forest (Shi et al., 2016a). Several machine learning approaches performed well, but it usually requires handcrafted features and structured label data in a large amount. For SVM-based methods, handcrafted features are utilized to differentiate between crack and non-crack images. These handcrafted features in SVM cannot conduct hierarchical processing on the input features. The raw data available for processing through machine learning approaches cannot generate semantic features of cracks. Other than ML techniques, deep convolutional neural networks (DCNN) are also utilized for detecting crack through classification (Mandal et al., 2018), localization (Y. J. Cha et al., 2017), and segmentation (D. Lee et al., 2019).

The CNN architecture automatically extracts features from the training examples which discriminate CNN from traditional ML techniques. Figure 1.1 shows the images

are classified as crack images or non-crack images. The localization determines the coordinates of cracks inside the image and builds bounding boxes around the cracks for detection, as shown in, Figure 1.2. For pixel-level segmentation, each pixel is classified as crack pixels or non-crack pixels as shown in Figure 1.3.

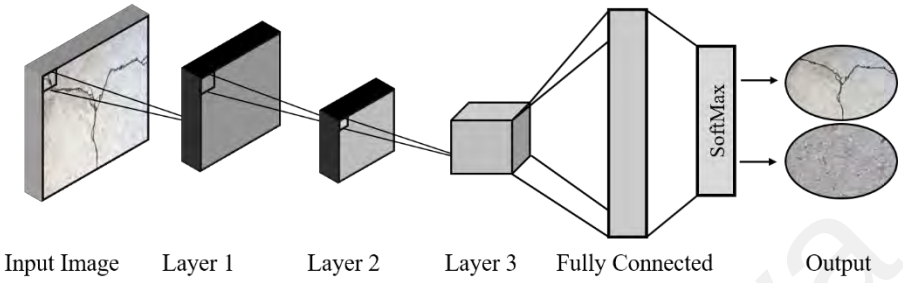

Figure 1.1: Crack image classification

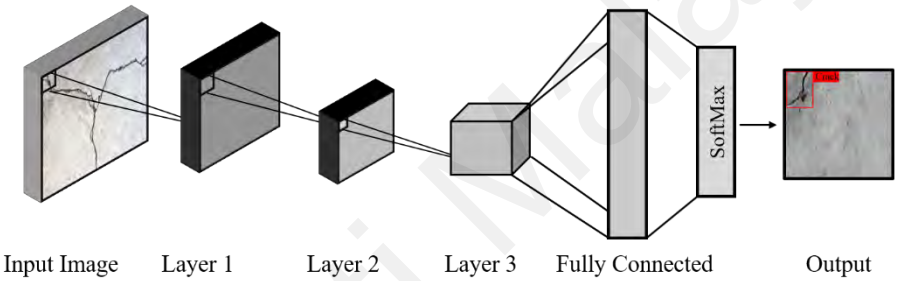

Figure 1.2: Crack bounding box

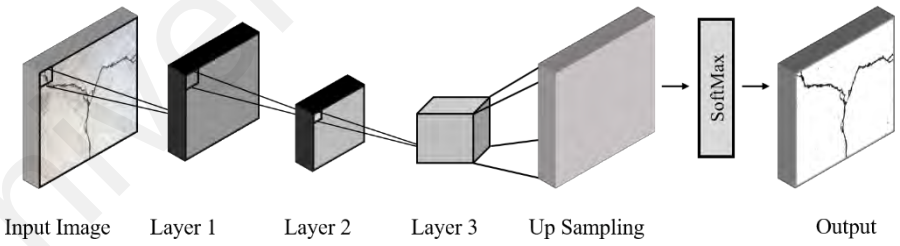

Figure 1.3: Crack pixel segmentation

Crack classification and bounding-box detection are useful; however, these techniques are not helpful to specify the accurate shape, size, orientation, and localization information of the crack region in the given image. On the other hand, semantic segmentation provides an instinct differentiation between crack pixels and non-crack pixels which is helpful to extract contextual crack information.

1.2 Problem Statement

Although the performance of semantic segmentation is exceptional, due to various reasons such as low-level features of crack pixels and the class imbalance problem in deep neural networks, the overall performance and crack pixel segmentation accuracy is significantly degraded. Crack pixels possess low-level features that are extracted and used for pixel classification. These crack features have a very similar appearance but trifling nature with non-crack pixels due to texture, pixel intensities, irregularities, inhomogeneous illumination, objects, and obstacles. The two segments are characterized by crack pixels (minority class) and non-crack pixels (majority class). The proportion between crack and non-crack pixels varies from image to image. Similarly, different datasets have different proportions between these two classes. In CrackForest Dataset (CFD) (Shi et al., 2016c), 98.4% of pixels are non-crack pixels and 1.6% are crack pixels for 118 images, whereas, in the DeepCrack dataset (Y. Liu et al., 2019) 96.5% pixels are non-crack pixels and 3.5% are crack pixels for 537 images. The non-crack region has a higher number of pixels as compared to the crack region as shown in Figure 1.4. The distribution of crack and non-crack pixel in Figure 1.4 are 37.1%/62.9%, 4.1%/95.9%, 5.6%/94.4% and 0.6%/99.4%, respectively. The pixels corresponding to a majority class tend to have a large variation in the number of pixels as compared to minority class pixels. The majority pixels have more influence on the training process, the trained network becomes biased and tends to classify the pixels as of majority class (Sudre et al., 2017). This variation in pixel numbers between two classes leads to the false prediction of a crack pixel in the testing process and the cumulative loss overwhelms the final loss. The low-level features along with imbalanced data decrease the performance of the trained network and make the learning process inefficient to distinguish between semantic relationships from the data (Shuai Li et al., 2018). Therefore, it becomes a scheming task to differentiate between the crack pixels and non-crack pixels.

1.2.1 Problem Due to Class Imbalance Dataset

The crack image is categorized into two segments, i.e., crack pixels (minority class) and non-crack pixels (majority class). The non-crack region has a higher number of pixels as compared to the crack region. The pixels corresponding to a majority class tend to have more pixels compared to minority class pixels. Therefore, the pixels belonging to the majority class influence the training process; the trained network becomes biased and tends to classify the pixels as of the majority class (Vluymans, 2019). This variation in pixel numbers between two classes leads to the false prediction of a crack pixel in the testing process, and the cumulative loss overwhelms the final loss. The low-level features and imbalanced data decrease the trained network's performance and make the learning process inefficient to distinguish between semantic relationships from the data (Cui et al., 2019). It becomes a scheming task to differentiate between the crack pixels and non-crack pixels. In (M. Wang & Cheng, 2019), the authors used the cross-entropy loss function to determine the error between ground truth labels and predicted labels. Furthermore, in (Ji et al., 2018) binary cross-entropy loss function is used to measure the difference between predicted and ground truth labels. A median frequency class weight with cross-entropy loss is implemented (X. Zhang et al., 2019) to balance the class pixel ratio. A hybrid loss function is proposed by (Wenjun Liu et al., 2019) by merging binary cross-entropy loss with a dice loss function.

Figure 1.4: Crack and non-crack pixels

To segment the crack and non-crack pixels accurately, the cross-entropy loss function operates on per pixel evaluation. Subsequently, this cost function evaluates each class prediction for every individual pixel and calculates the average loss for all pixels. This causes the majority class to be more dominant which leads to an increase in false predictions. Therefore, the usual approach using only cross-entropy loss is sensitive to the imbalanced dataset. Furthermore, the weighted cross-entropy loss function gives extra weight to the minority class, but the optimal weight value is hard to determine. The median frequency or inverse frequency weights are used as weighting factors for cross-entropy loss function. These weighting factors are global values that are calculated by the ratio of the total number of crack pixels and non-crack pixels in the entire dataset. In any given image, the crack region is different in size, width, and orientation which causes the difference in ratio between crack pixels and non-crack pixels. The dice function determines the overlapping area between two samples but initially, the dice coefficient is near 0, which causes instability during the training process (Abraham & Khan, 2019). Furthermore, it does not consider the background class for segmentation and ignores the smaller segments. Dice coefficient weights equally for both false positive and false negative cause low recall value. Therefore, a method to balance the difference between crack pixels and non-crack pixels is required.

1.2.2 Problem Due to Similar Features

Other than the two main regions (crack and non-crack regions), there are some other regions in the crack image such as stains, spots, blurring portions, poor continuity, low contrast, and object obstacles (Y. Liu et al., 2019). These regions are considered as noises that have some similar features to crack regions. The crack region possesses low-level features, edge information, and unique pixel intensity. These features are extracted and used for pixel classification. These crack features also appear similar in appearance but trifling in nature with non-crack pixels due to texture, irregularities, and inhomogeneous

illumination (Song et al., 2020). Therefore, a method is required to exclusively learn the low-level features of both regions and generate an output mask with high segmentation accuracy.

1.2.3 Problem Due to Sensitive Pixels

Boundary pixels in semantic segmentation are considered the hardest pixels to predict because there is no specific rule which differentiates the pixels in the transit region between two objects. Similarly, the boundary pixels between crack and non-crack regions are also difficult to distinguish (Cheng et al., 2018). Other than boundary pixels are considered sensitive pixels which are challenging to predict correctly. These pixels are difficult to predict because, during the training process, some of these pixels are classified as crack pixels when the intensities of these pixels are closer to crack pixels. Moreover, if these pixels' intensity is near to non-crack pixels, the network learns these pixels as non-crack pixels. Therefore, an efficient method is required to classify sensitive pixels accurately.

1.2.4 Crack Characteristics Measurement Problem

Crack characteristics such as length, width, and orientation are useful indicators to analyze the severity of the crack and the strength of the structure. These indicators help to determine the condition of exposure so that the appropriate action can be carried out to ensure the safety of the structure. Several image processing-based methods such as morphological operations, median axis algorithm, and Euclidean methods have been proposed to measure the length, width, and area of the crack in the given image. These methods shrink the crack into its skeleton which does not provide the precise measurement of crack characteristics. Further, the width of the crack varies at each point and cannot be represented with a single value.

1.3 Research Objectives

This study aims to propose a pragmatic and efficient crack pixel segmentation method that accurately classifies each pixel in the image as a crack or non-crack pixel. The following objectives are proposed for this research to achieve the aim of this study;

1) To investigate the convolutional neural network-based crack detection methods used for crack image segmentation.
2) To build a crack detection system using a convolutional neural network that generates a crack segmentation mask with improved segmentation accuracy through proposed network architecture and loss function that minimizes the effect of an imbalanced dataset, spots, stains and dark intensity regions.
3) To evaluate the performance of the proposed crack detection system by comparing the results with current models.

1.4 Research Methodology

The research methodology adopted to achieve the objectives of this research work is highlighted in this section, as shown in Figure 1.5.

1) A review of previous research on crack detection is carried out to find the effective techniques proposed and implemented for crack detection.
2) The issue associated with crack segmentation using an imbalanced dataset is highlighted based on the literature review.
3) Study different CNN architectures that have been proposed for the crack classification and segmentation to develop an optimal network for this research.
4) Study different loss functions to understand their capabilities and constraints for tackling imbalanced datasets and build weighting factors to overcome the stated issues.

5) Collection of crack image dataset such that the image contains spot, stain, and dark intensity regions, is highly imbalanced in terms of crack and non-crack pixels. Later, the dataset should be publicly available for other researchers to use the dataset for research purposes along with ground truth information.
6) Implementation of proposed network architecture and loss function for crack and non-crack pixel accuracy for imbalanced dataset.
7) To show the effectiveness of the proposed network and loss function, the results are compared with other network architectures and loss functions.

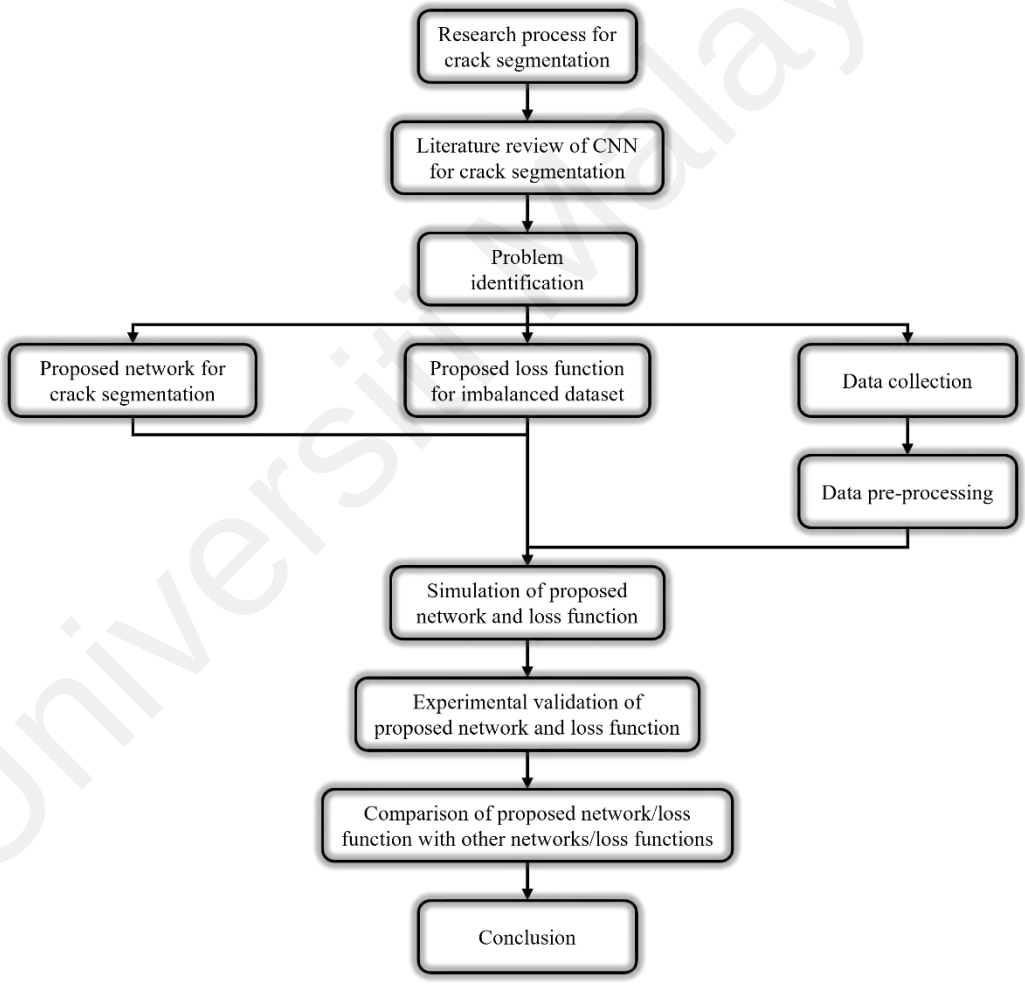

Figure 1.5: Methodology of research process

1.5 Scope of Research

Semantic segmentation is an interesting computer vision problem, which classifies each pixel of an image into its respective class or region. It generates a labeled mask as an output that specifies the region of a target object and background object separately. The labeled regions provide the precise localization information of each object. A DCNN is the most recent and successful technique used for pixel classification. Several DL architectures have been proposed and deployed to precisely classify the pixel of different regions. Crack region segmentation is a binary classification task, which classifies the pixels of a crack image into two classes i.e., crack pixels and non-crack pixels. The different number of crack and non-crack pixels in any given image introduces a class imbalanced data problem which results in low segmentation accuracy. Moreover, the boundary pixels and dark intensity regions are hard to predict accurately because they possess dual properties of both crack and non-crack regions.

Therefore, this study focuses on critically analyzing the effect of class imbalance data in the training process. It investigates the potential of DL architectures and other methods to balance the natural difference between crack and non-crack pixels. A balancing factor is generated from the ratio of crack pixels and non-crack pixels. Besides, a sensitive transform map is incorporated in the loss function to precisely classify the pixels of boundary and dark regions.

In addition, the second part involves the modification of residual blocks in the encoder-decoder network. The crack segmentation model is proposed for pixel classification. The proposed crack segmentation network (CSN) contains skip connections and residual blocks to recognize the crack and non-crack pixels in their respective regions. Finally, multiple experiments are conducted to evaluate the performance of the segmentation network and validate the overall system robustness.

In summary, the proposed crack segmentation methods have great potential not only in the crack detection field, but also in other fields such as object segmentation, medical images segmentation, and so on.

1.6 Thesis Outline

This thesis consists of five chapters. **Chapter 1** presents the overview of the structural health monitoring system, problem statement, research objective, and scope of the research.

Chapter 2 presents a concise and comprehensive literature review on machine learning, deep learning, convolutional neural network, network architectures, structural health monitoring system, and existing research.

Chapter 3 describes the methodology of the proposed solution, including network architecture, loss function, transform map, system configuration, and hardware description.

Chapter 4 evaluates the experimental results obtained from the simulations and the performance of the proposed method. The performance of the proposed method is further analyzed in comparison to existing research.

Chapter 5 summarizes the research work, the current limitation of the proposed work, and proposes the directions for further improvement in the future.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

Computer vision (CV) makes a machine capable of learning from the features of digital images and videos. It provides a better understanding of features and patterns through visual information. CV is interdisciplinary research, which involves engineering, computer science, mathematics, medicine, astronomy, transportation, agriculture, etc. A huge amount of visual data (smartphones and digital cameras) is available for these research fields. Image classification and segmentation are the most important problems in CV and digital image processing. Image classification differentiates between multiple objects from the image by extracting the unique features of that object while segmentation involves the partition of an image into several regions that possess different attributes (color, intensity, or texture) and objects based on the features of those objects. It aims to identify the background and objects in the foreground. Image classification and segmentation are applicable through multiple techniques, e.g. object detection, localization, or recognition tasks in many applications such as face recognition (Coskun et al., 2017) (Kute et al., 2019) (Peng et al., 2018) (Vo et al., 2018), disease diagnoses (Albarqouni et al., 2016; Lu et al., 2018; naceur et al., 2018; Vinícius dos Santos Ferreira et al., 2018), agriculture (Dias et al., 2018; Lottes et al., 2018; Razavi & Yalcin, 2017; Sobayo et al., 2018), the Intelligent Transportation System (ITS) (Gundimeda et al., 2019; Nugraha et al., 2018; Soon et al., 2019; Xinchen Wang et al., 2019), wireless communication (Dörner et al., 2018; Hadhrami et al., 2018; Mao et al., 2017), and cybersecurity (W. H. Lin et al., 2018).

2.2 Machine Learning

In 1959, the first research to verify programmable computing concluded that the machine could learn to defeat a human being in a checker game (Samuel, 1959). As shown in Figure 2.1, ML is a subfield of artificial intelligence (AI) and DL is also known as

representation learning, considered one of the most popular ML algorithms. The concept of ML has got many definitions based on the thoughts and perceptions of different scientists and researchers. The most referenced explanation of ML by Mitchell (Mitchell, 1997) is "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". ML is becoming an attractive and mature method that help to solve many real-world problems. it is devised that ML can be applied to many other real-life problems. It is the science that capable computers to perform well in solving problems without being taught each and everything. It is so common and adaptive today that without knowing we use it in many ways in daily life. ML is a powerful tool that predicts the outcome for a given input with the help of statistical, stochastic methods, AI algorithms, and advanced digital computing. It requires a huge amount of data for processing, hardware for computational performance (GPU), training techniques (activation, normalization, dropout), and advanced networks (CNN). These important processes are required to learn and perform the tasks effortlessly, intelligently, and efficiently like humans (Goodfellow et al., 2016). Most of the latest DL methods have produced promising results over different tasks i.e., classification (object, action, words, email, news) (Dyrmann et al., 2016; LeCun, 1989), segmentation (objects, scene) (Y. Xu et al., 2019), recognition (face, pattern, speech, object) (Coskun et al., 2017; Fukushima, 1988; In Jung Kim & Xie, 2014), detection (face, object, fraud, behavior, emotion) (Nugraha et al., 2017), identification (writing, gender, age, language) (Bhatt et al., 2017; Socher et al., 2011), sentiment analysis (emotions from text or speech) (Hassan & Mahmood, 2017), prediction (weather, traffic) (Salaken et al., 2019), social media services (suggestion, possible reaction, people you know) (Hayat et al., 2019), medical services (detecting disease, identifying infected area) (C. Chen et al., 2021; Fourcade & Khonsari, 2019), recommendation (products and services) (S. Zhang et al., 2019), robots

(chat-bot, virtual personal assistant) (Nuruzzaman & Hussain, 2018), machine translation (Costajussà, 2018), and surveillance (Pan et al., 2018; Sreenu & Saleem Durai, 2019). The learning mainly based on a mathematical model, criteria for training the model and optimization procedure. The learning process learns the features from the given dataset where learning procedure is categorized into supervised, unsupervised, semi-supervised learning and reinforcement learning algorithms.

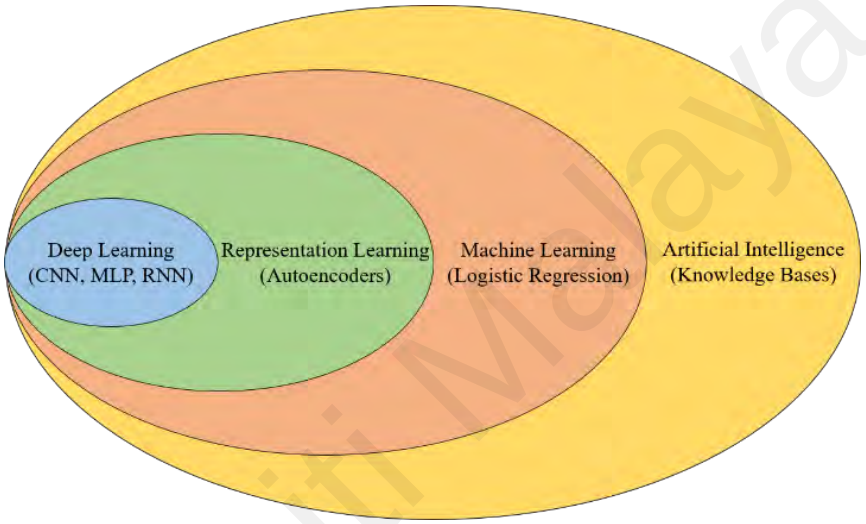

Figure 2.1: Venn diagram of AI and its sub-divisions (Goodfellow et al., 2016)

2.2.1 Supervised Learning

In supervised learning (Courville, Ian Goodfellow, 2016), algorithms are trained over a N number of paired sample data $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots \dots \dots (x_N, y_N)\}$ and the machine is known to input $X = \{x_1, x_2, x_3, \dots x_N\}$ and its labelled output $= \{y_1, y_2, y_3, \dots y_N\}$. The leaning algorithm develop a function $f(x, y)$ from sample data which is mostly independent and identically distributed used for prediction, classification, and regression-related problems.

2.2.2 Unsupervised Learning

The unsupervised learning algorithms are different from the supervised one as it does not have input-output pair rather they have unlabeled data. The algorithm tries to extract hidden features and builds a statistical model from the given data in the form of a group or clusters. The other category of the ML algorithm is reinforcement learning which has a feedback loop between the learning procedure and the output of trained data based on error.

2.3 Deep Learning

DL can be defined as an ML algorithm that deals with neural networks. Neural networks with a deep structure or with more than two layers are referred to as Deep Neural Networks (DNN). Figure 2.1 shows that DL represents representation learning which is a sub-type of ML that consists of multiple levels of representation (Heaton, 2018). DL over the past several years has been developed as a popular tool that attracts the attention of researchers from other fields to overcome the weaknesses of traditional methods and solve complex problems in their respective fields to achieve demanding results. The popularity of DL is doable due to the availability of a large amount of data, computational performance due to graphical processing units (GPU), training techniques such as Adam (Kingma & Ba, 2014) and SGDM (Rumelhart et al., 1986), optimized layers such as ReLu and SoftMax, and advance networks such as CNN (Krizhevsky & Hinton, 2012) and U-Net (Ronneberger et al., 2015). With the increase in databases, DL has exponentially achieved success both in commercial and academia. Not only did software base advancement help DL to achieve success, the latest hardware such as GPU's improved the ability of DL (Courville, Ian Goodfellow, 2016). DL with deeper layers improves the system's experience by learning the features from data and making complex structures deeper and simple (Patterson & Gibson, 2017). Therefore, it is a novel discovery for solving problems in those areas which has high-dimensional data. Inspired by brain

function, deep neural networks are built from many hidden layers sandwiched between the input and output layers. DL provides significant techniques to solve classification problems (X. Glorot et al., 2011; Hu et al., 2015; Y. Jang et al., 2018; I.J. Kim & Xie, 2015; H. Lee et al., 2009; H. Zeng et al., 2018), object segmentation (Girshick et al., 2014; J. Long et al., 2015), object detection (Girshick, 2015; Peng et al., 2018), regression (Salaken et al., 2019), and natural language processing (Jung, 2019). DL is not a task-specific learning approach. It can be successfully applied in several research domains as a universal learning method, capable to provide a solution in almost every field.

DL techniques can be classified into a CNN (Rawat & Wang, 2017), restricted Boltzmann machine (RBM) (H. Chen & Murray, 2003), autoencoder based coding, and sparse coding methods. These techniques have developed several other ideas such as deep belief network (Hinton, G. E. et al., 2006), recurrent neural networks (RNN) including long short term memory (LSTM) (Gers et al., 2002), gated recurrent units (GRU) (Chung et al., 2014), recursive neural networks (RNN) (Socher et al., 2011) and recent generative adversarial network (GAN). The selection of DL type depends on the scope of the problem.

2.3.1 Convolutional Neural Networks Overview

DL approaches are excellent in solving traditional artificial intelligence-related problems. The most established, progressive, and a widely used algorithm is CNN. The following section discusses in detail the CNN, its variants, and applications.

2.3.2 CNN Layers

The most established, progressive, and widely used DL algorithm is the CNN (Krizhevsky & Hinton, 2012), developed in the 1980s (Fukushima, 1988). The idea did not attract researchers due to the absence of the computational ability of hardware, high processing machines, and large storage devices. However, the concept accelerated as

machines' processing increased in terms of computation and database to retrieve and store (Russakovsky et al., 2014). Later in (LeCun, 1989), CNN's were successfully applied in classification problems and performed brilliantly in CV applications. CNN architecture is divided into two divisions: feature extractors and classifiers as shown in Figure 2.2. The feature extractor consists of a convolutional layer, pooling layer, and rectifier unit which extracts all the lower features from input data. The classifier consists of two parts, a fully connected layer and an object classifier for identification. The input image convolves with the kernel (learnable filter) which slides over the input image to extract the features. The result of convolution is then passed through the pooling layer (mostly max pooling) and the output of the pooling layer goes through the nonlinear activation function (mostly sigmoid function) which extracts more features. The output is passed on and follows the same procedure to extract more and more features.

The starting layers extract edges and patterns, which are inferior features. The middle layers extract features like the shape of the object and the color of the object, whereas the higher layers extract features such as complete objects. The low-level features propagate towards higher layers where higher-level features are extracted to be used as an input for the fully connected layer to generate output. The final output from the features extractor layer enters into a fully connected neural network (Arel et al., 2010) for classification, bounding box, and pixel classification layer for segmentation. The classification is done using a classifier such as the Soft-Max function. The network parameters determine the size of output i.e., depth, stride, and padding. CNN is developed as the most widespread and successful DL architecture for several types of input data. It belongs to a feed-forward DNN class, which has several state-of-the-art architectures. The CNN compresses the fully connected network (FCN) by lessening the connections and sharing the weight of the edges.

Convolution performs 3 main tasks: sparse interaction, parameter sharing, and equivariant representation.

i) **Sparse Interaction:** In a neural network, every output unit interacts with every input having separate parameters. These parameters help to determine the relationship or interaction between the input and output unit. The CNN uses kernels of different sizes which are smaller than the input data in size. This reduces the number of learning parameters, the storage space and increases computation efficiency.

ii) **Parameter Sharing:** It uses the same parameter for more than one chunk. In convolution, each kernel value is used at every point of input other than boundary values. It helps CNN to use only one set instead of multiple parameters for every location. It reduces the storage requirement further.

iii) **Equivariance:** It refers to the shift in the feature map by the same amount as input shifts. Convolution does the same but not naturally (Heaton, 2018).

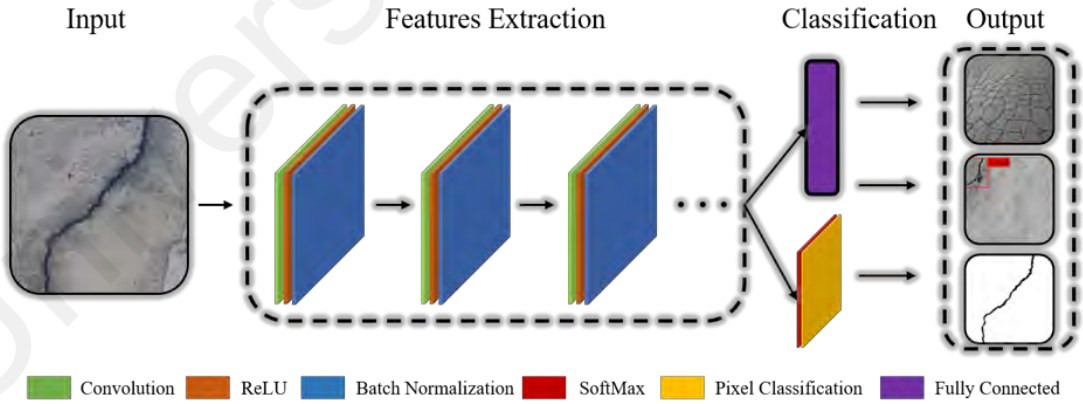

Figure 2.2: CNN architecture

CNN performed far better than other multi-layered perceptrons (MLPs). The CNN weights are shared and do not need to learn again for the same object at different locations. It recognizes visual patterns, directly from raw image pixels. This decreases the number of learnable parameters and has minimized the pre-processing task. The backpropagation learning method improved the performance by providing the

solution to deal with non-linearity with the decrease in the computation process due to a smaller number of weights.

2.3.2.1 Input Layer

The input layer is the first layer of CNN architecture that contains the data in the form of images. This layer understands the input data and represents the images as a pixel matrix. It gives the contents of input data and has no learnable parameters. The images are pre-processed in terms of dimension and size before feeding to this layer.

2.3.2.2 Convolutional Layer

The convolutional layer performs convolution operation which is the trademark of CNN architecture. This layer holds learnable parameters such as weights and biases. This layer contains filters or kernels, used to detect edges, shapes, and patterns of the given input image. Kernels are convolved with each input feature/image pixel to produce feature maps as an output. A dot product between each input and filter is performed, followed by summing each dot product output, and finally, a bias is added. Bias can be configured according to network requirements. The convolutional layer reduces the computational cost by minimizing the input size.

$$Z_i^l = \sum_{J=1}^{K^{l-1}} W_{i,j}^l * Z_j^{l-1} + B^l \tag{2.1}$$

The kernel computes the product of weight and input of kernel size. It also determines the desired features based on kernel weights. Equation (2.1) shows the operation of the convolutional layer where Z_i^l and Z_i^{l-1} are the outputs of current layer and previous layer respectively. $W_{i,j}^l$ and B^l represents filters and biases. Each neuron need not be connected to all other neurons in the preceding and following layer. The input is convolved with filters to produce an output where bias is added for none zero value. The final output goes

through a non-linear activation function which activates the feature maps and forwarded the result to the next layer.

2.3.2.3 Pooling Layer

The other name of this layer is the subsampling layer. This layer reduces the dimensions through a down sampling operation. Average (uses the average value) and Max (uses the highest value) pooling are the two most used operations. The following Equation (2.2) of the subsampling function represents a pooling operation.

$$Z_i^l = Sub - Sampling\ (Z_j^{l-1}) \tag{2.2}$$

2.3.2.4 Nonlinearity Layer

It applies the relevant nonlinear activation function. The most common functions are sigmoid, rectified linear unit (ReLU), and SoftMax.

$$a^l = f(Z^l) \tag{2.3}$$

2.3.2.5 Fully Connected Layer

This is a flattened layer with each neuron of the previous layer connected to each neuron of the current layer. Each neuron has a separate weight for each connection. This layer has the highest number of learnable parameters. The input data is linearly processed, passed through a non-linearity, and then propagated to the next layer.

2.3.2.6 Loss Functions

Loss function plays an important role in convolutional networks. For the optimal result, the selection of an appropriate loss function is critical. The loss function minimizes the difference and maximizes the similarities between predicted labels and ground truth labels. It provides the optimized parameter values by calculating the difference between predicted and actual output. The difference is propagated back into the network, which helps update the weights of parameters and minimize the difference. It is required from

the loss function to generate maximum value for bad prediction and minimum value for good prediction.

Crack images contain highly imbalanced classes that cause serious issues in evaluating segmentation results. The distribution of crack and non-crack pixels is not even which causes an imbalanced data problem, because in maximum images the number of crack pixels is more than non-crack pixels. The imbalanced data causes the learning process to get trapped in the local minima of the loss function and makes the network biased. In recent years, multiple techniques are used to tackle imbalanced datasets. These techniques are divided into two categories (Vluymans, 2019); data level and algorithm level. The data level approach tries to balance the class ratio in a given dataset, whereas the algorithm level approach adjusts the learning algorithm or classifier to facilitate the imbalanced dataset. To handle this problem one of the methods is selecting the optimal loss function. Loss functions are distinguished from each other based on distribution, region, and boundaries (Ma, 2020). The following loss functions are used for crack image classification and crack pixel classification.

(a) ***Cross-Entropy Loss Function***

The cross-entropy (CE) loss function is mainly used for cost function in the training process of neural networks. It measures the difference between two random variables. CE loss is defined as

$$\text{CE\_Loss} = -((t\,log(p)) + (1-t)log\,(1-p)) \tag{2.4}$$

where t is ground truth labels and p is the corresponding predicted labels. CE loss has achieved excellent results on various CV tasks such as crack image classification (Silva & Lucena, 2018) and damage detection (Y. J. Cha et al., 2017). However, it has not succeeded in segmentation applications, especially where the data distribution is

imbalanced. Cross entropy loss evaluates each pixel prediction individually and averages them to get over all pixel predictions. Thus, it asserts equal learning to every pixel in the image and makes CE loss unsuitable for segmentation with an imbalanced dataset.

(b) ***Weighted Cross-Entropy Loss Function***

Weighted cross-entropy (WCE) loss is an extended version of CE. WCE assigns weight (β) to one class. In (X. Zhang et al., 2019), DCNN adopted weighted cross-entropy loss by applying median frequency class weights to balance pixel classes and is given by

$$\text{WCE\_Loss} = -(\beta(\ (t\ log(p))\ +\ (1-t) log\ (1-p)) \tag{2.5}$$

β is a weight for each class that penalizes the majority classes.

Another variant of the CE loss function is the Balanced Cross-Entropy loss (BCE), like WCE. The only difference is that BCE assigns weight to both classes. The weighting factor β for one class type and 1- β for the second class. In (Cheng et al., 2018), the author adopted U-Net architecture for semantic segmentation and deployed BCE loss with chamfer distance to handle the imbalanced dataset issue between crack and non-crack pixels.

$$\text{BCE\_Loss} = -((\beta\ t\ log(p))\ + (1-\beta)(1-t) lo\ g(1-p)) \tag{2.6}$$

Assigning weights through inverse class frequency on real-world applications has not reached acceptable accuracy on highly imbalanced datasets.

(c) ***Dice Loss Function***

Based on the dice coefficient, an optimized loss function is designed to segment the medical images (Milletari et al., 2016) which has a highly imbalanced dataset. It makes the network favor one class and ignores the other important class. It has two versions; the generalized dice loss is used for multi-class segmentation and the weight of each level is inversely proportional to class frequencies.

$$\text{Generalized\_Dice\_Coefficient\_Loss} = 1 - \frac{2\,tp}{t + p} \tag{2.7}$$

$$\text{Dice\_Coefficient\_Loss} = 1 - \frac{2\,tp}{t^2 + p^2} \tag{2.8}$$

(d) ***Tversky Loss Function***

Tversky loss (TI) (Salehi et al., 2017), is based on the Tversky index which is a

$$\text{Tversky\_Loss} = \frac{tp}{tp + \beta(1 - t\,)p + (1 - \beta)t(1 - p)} \tag{2.9}$$

generalization of Dice's coefficient. It adds weight to FP (false positives) and FN (false negatives) to address imbalanced data issues in segmentation tasks through a trade-off between precision and recall.

(e) ***Focal Loss Function***

Focal Loss (FL) (T. Y. Lin et al., 2020), adapts the standards of CE in which probabilities are multiplied by their original loss values, to down-weights the negative class so that the positive class gets more attention from the network. FL assists the network to focus on poorly well-trained pixels, therefore it performs well on an extremely imbalanced dataset. Crack segmentation based on a single stage detector (Carr et al., 2018), used the focal loss function which focuses on the outliers and misclassified samples by assigning higher weights to them.

$$\text{Focal\_Loss} = -((\beta(1 - p)^{\gamma} t\, log(p)) + (1 - \beta)(p)^{\gamma}(1 - t)\, log(1 - p)) \tag{2.10}$$

where γ is the steering factor that decreases the loss with its increase. Factor $1 - p$ lowers the CE version using alpha and gamma.

2.3.3 CNN Architectures

CNN is developed as the most widespread and successful DL architecture for several types of input data. It belongs to the feed-forward DNN class, which has several state-of-the-art architectures. Each architecture has a domain where they perform outclass as compared to traditional methods. These architectures contain several repetitive layers, a convolutional layer, a pooling layer, and an activation layer.

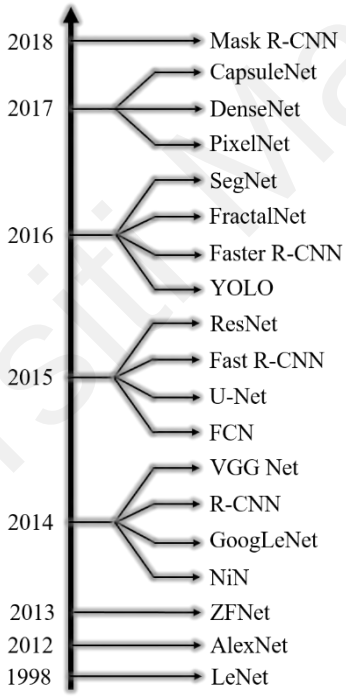

Figure 2.3: CNN history

Figure 2.3 contrasted the most common and popular, state-of-the-art CNN architectures which include LeNet (Lecun et al., 1998), AlexNet (Krizhevsky & Hinton, 2012), ZFNet (Zeiler & Fergus, 2013), NiN (M. Lin et al., 2013), GoogLeNet (Szegedy et al., 2015), R-CNN (Girshick et al., 2014), VGG Net (Vo et al., 2018), InceptionV3 (Szegedy et al., 2016), FCN (J. Long et al., 2015), U-Net (Ronneberger et al., 2015), Fast R-CNN (Girshick, 2015), ResNet (He et al., 2016b), YOLO (Redmon et al., 2016), Faster

R-CNN (S. Ren et al., 2017), FractalNet (Larsson et al., 2016), SegNet (Badrinarayanan et al., 2017), PixelNet (Bansal et al., 2017), DenseNet (Mocsari & Stone, 1978), CapsuleNet (Sabour et al., 2017), and Mask R-CNN (He et al., 2017).

CNN architectures such as AlexNet (Krizhevsky & Hinton, 2012), VGG Net (Vo et al., 2018), GoogleNet (Szegedy et al., 2015), R-CNN (Girshick et al., 2014), Mask R-CNN (He et al., 2017), InceptionV3 (Szegedy et al., 2016), and ResNet (He et al., 2016b) are fruitful when trained on a large image dataset, and have achieved the state-of-the-art results for general image classification and segmentation. These architectures have shown significant achievement in ImageNet Challenge (Deng et al., 2015). This challenge provides a platform to launch a benchmark in DL for classification, detection, localization, and segmentation tasks. The event allowed several algorithms to participate and identify the given image. The classification task has more than 1000 different objects and more than one million training images. The yearly performance of CNN-based architectures in the ImageNet Classification challenge is shown in Figure 2.4.

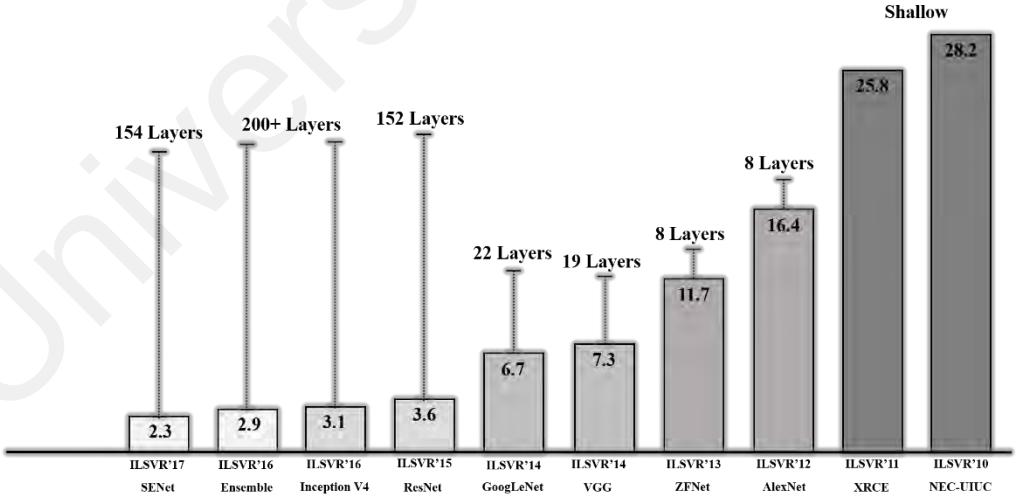

Figure 2.4: ImageNet challenge

2.4 Structural Health Monitoring System

Structural Health Monitoring (SHM) refers to a regular check-up mechanism to monitor the condition and characteristics of civil structures. SHM system is developed to monitor the state of health of a civil structure compared to normal conditions. Its primary

purpose is to identify the changes in the civil structure and devise a maintenance plan and take appropriate action against the abnormality in the structure (L. Long et al., 2020). However, there are many ways (Guo et al., 2017) to monitor structural health through multiple structure health indices. On the other hand, damage detection can be interpreted as a structural modification that changes physical properties and weakens the structure. A crack is generally defined as a defect that may cause serious damage and consequences to the structure. For the safety and maintenance of roads, subways, bridges, buildings, dams, tunnels, monuments, etc, cracks are one of the factors that determine the structural condition.

2.4.1 Cracks

In terms of image, cracks are sudden changes in the intensity of pixels. Cracks are thin dark lines that appear on the surface of solid material along which it has split without breaking apart. Cracks on any concrete surface can arise due to material shrinkage and expansion, shifting foundations, premature drying, overloading, hydrostatic pressure, unbalanced blend, swollen soil, poor soil bearing, creep damage, settlement, and farming.

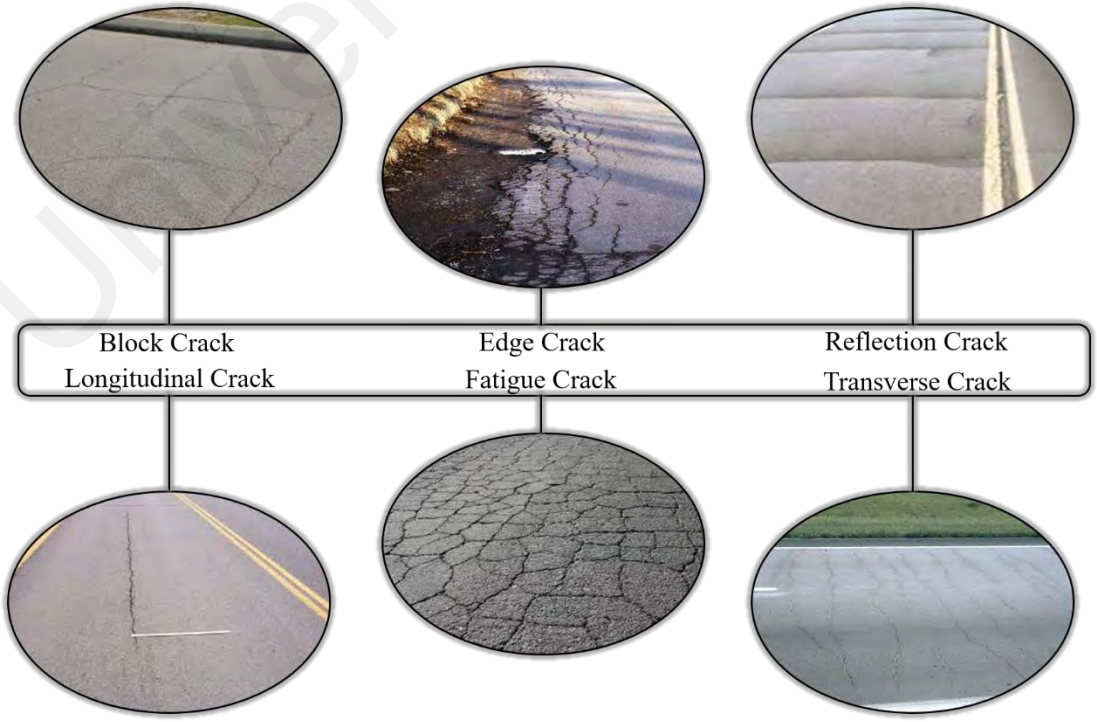

Figure 2.5: Types of crack

According to a report (John S. Miller & Bellinger, 2014), cracks are mainly categorized into fatigue crack, block crack, edge crack, longitudinal crack, transverse crack, and reflection crack, as shown in Figure 2.5.

Cracks can be visible to the human eye as a simple defect. However, the minor crack can turn into a bigger and more threatening fault. Some cracks are difficult to observe due to the complex texture pattern in the background. Different types of cracks have different sizes, like hairline cracks have 0.1 mm width and can be seen on the clear background but are challenging to observe with illumination variation. Fine cracks have a width of up to 1mm. Some cracks with a width up to 5mm are not considered as dangerous and can be healed. However, those cracks above 5mm in width can cause extensive damage and require proper repair works or replacement in some cases (M.-H. Zeng et al., 2020).

2.4.1.1 Crack Features

Features are the pieces of information that represent the identity of an object either for classification or detection. The system learns these features to understand the structure of an object and to differentiate between the objects. Differentiation is an easy task for human beings, but a machine may require many features to understand the uniqueness of an object. Feature extraction is a technique that transforms the data to a new dimension for a better representation of data. It is designed to extract discriminative features from the training data. The crack features are generated through feature extractors or kernels. These feature extractors have different orientations and sizes. At each level, these feature extractors find specific information regarding the length, width, shape, color, intensity, position, and location of the crack. Consequently, these extracted features are implemented as input in the learning algorithm. Image processing uses morphological operations such as erosion and dilation to extract crack pixel features from the image. Morphological operations also increase the contrast between crack and non-crack pixels.

These features include edge information, the contrast between foreground and background, and pixel intensities. ML uses handcrafted features to extract crack information from the image. For neural networks, hidden layers automatically extract feature information from the images. The features of an image are usually the values of the pixels in the image (Goodfellow et al., 2016). The CNN uses kernels to extract low-level crack features at initial layers and high-level crack features or complicated features at higher layers. At each level, these feature provides specific information such as edges, texture, corner, intensities, color, position, and location information of each pixel. The final layer differentiates between crack pixels and background pixels. These features can be classified into strong positive activation, strong negative activation, and neutral, which indicates the activation strength of each channel.

2.4.2 Crack Detection

Crack detection is a process of localization or detection of cracks located on any civil structures, manually by skilled labor or automatically by machines. A comprehensive literature study of image-based crack detection via deep CNN is carried out in this review article. According to our knowledge, the most recent articles on CNN and crack detection are comprehensively covered. The effective analysis of multiple factors such as CNN type, learning methods, computation complexity, dataset, image type, error estimation, results, loss functions, evaluation metrics, and hardware used has been presented. The next portion discusses manual crack detection, image processing techniques used for crack detection, and crack detection through machine learning.

2.4.2.1 Manual Crack Detection

Manual Crack detection is a combinational effort of several methods that help to determine the alarming sign (cracks) on civil structures. Cracks can be appearing on any civil structures including roads, subways, bridges, buildings, dams, monuments, etc.

Cracks are considered one of the initial indications which tell about the degradation of civil structures. For maintenance, crack detection is a crucial task and needs special attention. Periodical checking is needed to determine the condition of infrastructure. Early detection can help to prevent bigger disasters. It is also useful as it ensures the safety and estimates the life duration of the civil structure.

2.4.2.2 Limitation of Manual Crack Detection

Cracks can be detected by manual inspection or automatic inspection. Manual detection, which uses human resources (manual) requires skilled (experience and knowledge) labor with appropriate tools to regularly inspect and identify the cracks by following the safety precautions, which makes this activity very costly and time-consuming. This process also requires a lot of documentation as it does not involve any visual record and it is still hard to pass a judgment about the sensitivity of crack. In a large infrastructure, manual detection is a lengthy process and needs to cover a vast area.

2.4.2.3 Crack Detection using Image Processing Techniques

Techniques that root from image processing for crack detection mainly involve image pre-processing, segmentation, extraction of features from the image, and crack recognition. Image processing techniques use edge detection (scan the sudden change in pixel intensity of pixels) and segmentation (identify the object). It is one of the earliest techniques to detect, classify and segment the crack from visual data through captured images to solve the crack detection problem. Multiple techniques are used to classify cracks through image processing methods. From images, cracks can be detected using the edges information (Abdel-Qader et al., 2003), morphological operations (Yamaguchi et al., 2008), statistical methods, digital image correlation (Gehri et al., 2020), and pattern matching (Adhikari et al., 2014b). The process undergoes from capturing the image, processing, extracting crack features, and identifying the crack. Crack pixels are assumed

to be darker in intensity as compared to other nearby pixels. The contrast between neighboring pixels of an image is compared to classify the crack and non-crack regions (B. Xu & Huang, 2005). Based on pixel intensity, a threshold is set to classify between crack and non-crack pixels using statistical methods (Tsai et al., 2009). Different intensities are used to determine the pixel's probability to classify as a crack or non-crack pixel (Zou et al., 2012). A three-stepper procedure based on mathematical morphology and curvature evaluation is developed. Morphology enhances the image quality with cracks, and then a curvature operation is performed. The last image is passed through a linear filter to distinguish the cracks from the analogous background (Iyer & Sinha, 2005). A crack quantification method of 2D image (grayscale scanning electron microscope) is developed. The black and white image is separated, filtered, and quantified to enhance via five morphological operations. Connected components label cracks and determine the width, length, area, aspect ratio, and orientation (Arena et al., 2014). The extraction of statistical properties from pavement images is also used to segment the crack images (Koutsopoulos & Downey, 2006). For the maintenance and reparation of roads, the crack detection and classification method are presented (Cubero-Fernandez et al., 2017) which first degrade noise from the image and then highlight the cracks. After learning the features using a decision tree, the statistical classifier is applied to detect cracks in road images. The two-step crack detection method is designed based on statistical filtering. First, extracting the crack features and then segmenting the crack (Sinha & Fieguth, 2006). In (Koch et al., 2014), the graph-based method is used to locate the crack properties and later detect the crack, but this method requires the manual insertion of the starting and ending point of the crack. This problem was resolved by (Oh et al., 2009) when the proposed crack detection and tracing algorithm extracts crack properties for crack detection. Classification is also performed by the histogram method using a Support Vector Machine (SVM) to classify the crack and non-crack images (Prasanna et al.,

2012). Classification methods are based on binary tree and backpropagation, which divides the image into the crack and non-crack regions, comparing grayscale values. Otsu's method provides a good help in the segmentation of images (L. Li et al., 2014). The principle Component Analysis (PCA) algorithm, commonly used for dimensionality reduction, is also employed to determine cracks from the image set (Abdel-Qader et al., 2006). Filters are also used to detect cracks by combining the binary versions of the crack image. The original image is convolved with filters of different orientations (Salman et al., 2013). Based on a threshold value, background pixels are separated from foreground pixels, and Sobel's filter eliminates noise. Lastly, Otsu's method is applied to detect major cracks (Talab et al., 2016). A crack detection, MATLAB-based toolbox CrackIT (Oliveira & Correia, 2014) is proposed to detect and characterize fine cracks with at least 2mm width. Cracks in the concrete structure are detected by first converting the image into grayscale and then applying Sobel's filter for detecting the crack.

2.4.2.4 Limitation of Image Processing Crack Detection Techniques

For the safety of civil infrastructure, low-cost automatic crack detection is designed for maintenance purposes. Crack detecting through image processing techniques is not straightforward. Many environmental conditions such as shadow, dust, stain noise, multicolor spots, uneven light illumination, multiple background scenes, and change in the dataset can create resistance in detecting cracks from the image. While designing an algorithm, one must consider the characteristics of both the crack and background (rest of the image), including the camera's position, camera resolution, length of the crack, width of the crack, and angle of a crack in the captured image. A crack can be one or a few pixels wide edge in the image and a line with some width, which is generally darker than the background pixels. Different crack detection methods are divided into two classes based on features: low-level features and high-level features. These features have two portions: edge detection and image segmentation. Crack with good continuity and

has high contrast are easy to detect with high accuracy, but the problem in crack detection is noise in the background (poor continuity due to noise and low contrast due to shadow and direction of exposure) which leads to the degradation of methods, those are based on low-level features. Crack detection through image processing techniques is challenging because crack possesses poor continuity and low contrast among neighboring pixels. The problems with intensity and contrast-based algorithms include.

a) The change in orientation of the image affects the output.
b) 2-dimensional images do not possess in-depth information.
c) They lack a proper methodology that identifies the universal threshold.
d) The result fluctuates by varying the distance between source and camera.
e) For noisy images, these algorithms do not perform remarkably as compared to less noisy images.
f) Dark pixels in the image that are not part of crack lines can also be detected as a crack portion.
g) Some image-based algorithms of crack detection for final judgment require human inspection.
h) An algorithm can perform well for a particular dataset, yet its performance may still degrade for other datasets due to different conditions.
i) Few methods also require manual insertion of starting and ending point of crack, which requires extra effort in terms of processing and time.
j) The hand-crafted features are not robust and involve intensive computation. Therefore, these features are not enough to differentiate the crack and complex background, especially in the low-level image.

2.4.2.5 Crack Detection using Machine Learning Methods

Machine learning is a powerful tool that is widely implemented in almost every research field. The primary task of these methods is feature extraction and pointing out those areas in the image that contains cracks. The graph-cut segmentation method divides the image into the crack and background pixels to classify the image according to extracted features (Moussa & Hussain, 2011). For classification, an SVM can classify cracks of different orientations and compute geometric values of cracks. Another machine learning-based crack segmentation is used to remove background elements first and then features are computed based on color and texture. These features help to classify the crack images by training the SVM (Varadharajan et al., 2014). Other crack detection techniques based on ML methods involved Deep Belief Network (DBN) (Xuejun Wang & Zhang, 2017) for crack detection, simple classifier based road crack detection and characterization (Oliveira & Correia, 2013), Markov-based method (Delagnes & Barba, 2002), image binarization (Ahmadi et al., 2018b), random forest (Shi et al., 2016b), recurrent neural network-based crack detection in 3D asphalt surface (A. Zhang et al., 2019) and AdaBoost through texture pattern recognition (Cord & Chambon, 2012). These methods performed well on those images which have clear and visible cracks. The processing is based on feature extraction, so it becomes difficult and ineffective to extract crack features from unclear images. However, DL methods have shown good results as compared to traditional image processing-based methods and other machine learning-based methods.

2.4.2.6 Limitation of Machine Learning Crack Detection Techniques

Many ML methods performed well but have several limitations such as poor reusability of modules and requiring a large amount of structured, tagged, and hand-crafted data for the training process with no convincing results as compared to neural networks. In the previous section, most of the ML methods require many structured

labels. For complex nonlinear regression, it is challenging to formulate distinctive decisions at crack image boundaries. For SVM-based methods, handcrafted features are utilized to differentiate between crack and non-crack images. These handcrafted features in SVM cannot conduct hierarchical processing on the input features. Therefore, the raw data available for processing through ML approaches cannot generate semantic features of cracks.

2.4.2.7 Crack Detection using Deep Convolutional Neural Networks

DL can detect cracks through classification, localization, or segmentation. Images are labelled as crack or non-crack for classification. For pixel-level segmentation, pixels are classified as crack or non-crack pixels whereas localization determines the coordinates of cracks inside the image and builds a bounding box around the cracks for detection. Figure 2.6 shows the output form of classification, segmentation, and bounding box, respectively.

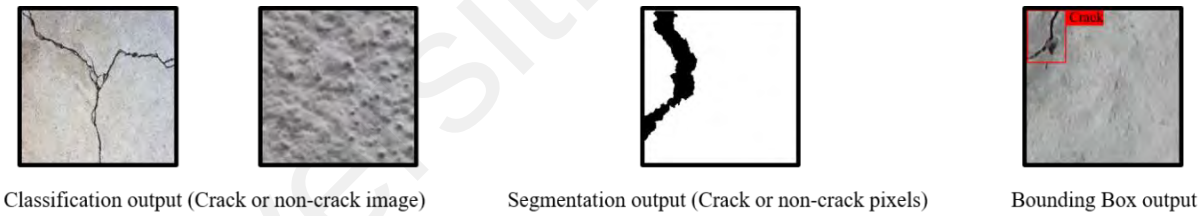

Figure 2.6: Output representation

2.5 Related Works

From the literature on crack detection using CNN, it is concluded that crack detection is divided into two groups. The first way of crack detection through CNN architecture is a sliding window technique and the second method is pixel-level crack detection. We investigated both parts separately and discussed the findings. DCNN can capture the non-linear and dynamic relationship between input and output. Hence, they can classify and segment the data by learning from the environment. The crack detection process identifies cracks either at the image level (classification) or at the pixel level (segmentation).

2.5.1 Crack Classification Techniques

CNN learns the detailed features and patterns of cracks from the images to perform the classification tasks. If the input image has a high resolution, then it requires a large storage capacity and processing power. Therefore, a large number of images are not recommended to insert at once into the network. The sliding window technique convolved the input image with kernels, which intends to present the image in a computable form or small group of pixels that then feeds to the network. The kernel is shifted one pixel right and selects the next patch of pixels to be inserted into the network. Thus, the process saves computational cost and processing. In the last few years, several crack classifications methods are proposed that use a sliding window technique. The classification-based methods for crack detection are discussed categorically in the following section.

2.5.1.1 Classification using DCNN

For safe driving, an automated road crack detection method is developed to detect the road cracks (Lei Zhang et al., 2016a), which learns the discriminative features from the image patches and classifies each input image patch as a crack or non-crack image. The network is trained over 500 images of 3,264 x 2,448 size using Stochastic Gradient Descent (SGD) algorithm. These images are collected at Temple University and augmented into 1,000,000 images. The network consists of four convolution layers, two fully connected layers, and four max pooling layers. The ReLU function is adopted for activation and the sigmoid function is used for classification. The designed model has achieved 86.96% precision, 92.51% recall, and 89.65% F1-Score, respectively. The model showed better performance as compared with the SVM and boosting technique in classifying the crack patches. This method used CNN as a classifier to predict a label from the local path, which is based on the local context of a crack pixel. The average probability is generated randomly around the center pixel c by,

$$p(c|\{P_1(c), \dots, P_N(c)\}) = \frac{1}{N}\sum_{i=1}^{N} P_i(c) \quad (2.11)$$

where $P_i(c)$ is the classification probability of ith patch. The ConvNet overestimates the crack probability which makes it difficult to mark crack pixels for segmentation. Therefore, this method does not provide a standard criterion that differentiates between positive and negative crack samples.

Crack detection techniques encounter several real-world challenges that can be minimized through DL methods. The effectiveness of a deeper network for pavement crack detection (Pauly et al., 2017) has been demonstrated by varying the network layers and measuring its effect on accuracy. The optimized proposed network classifies the input images as a crack or non-crack images. The dataset consists of 500 original images, divided into two subsets of 3,264 x 2,448 image size, collected at Temple University. The network consists of convolutional layers followed by activation layers, four pooling layers, a dropout layer between two fully connected layers, and a SoftMax classification layer. Two different networks are designed to analyze the performance, four convolutional layers are used in one network and five convolutional layers are used in the other network. Both networks are trained and tested with the same dataset, and later these trained networks are tested on different datasets. The results show that the increase in the convolutional layer produced better results and achieved 91.3% accuracy, 90.7% precision, and 92% recall, respectively. The second conclusion states that using different datasets for training and testing leads to degradation of performance.

The effect of varying illumination on the source image makes crack detection a challenging task. Vision-based concrete crack detection is developed to classify input images as crack or non-crack images (Y. J. Cha et al., 2017). This model extracts the cracks through CNN at different atmospheric conditions without measuring the defect

features. The dataset consists of 277 raw images of high resolution (4,928 x 3,264), augmented into 40,000 images of size 256 x 256. These images are inserted into the proposed CNN model for training. The architecture has four convolutional layers, two pooling layers, a batch normalization, a ReLU activation function, a dropout layer, and a SoftMax layer for prediction. For learning and training, stochastic gradient descent using backpropagation is applied. The CNN-based method is compared with other traditional methods (Canny and Sobel edge detectors). The results showed that DL-based algorithms performed by achieving 98.22% accuracy as compared to conventional algorithms. This method performed well but still cannot be considered reasonable and reliable for corner or edge cracks due to mainly two reasons. First, due to the shrinkage of the image as it goes from the first layer to the last layer, this method discarded those images that have cracks at corners or edges. Second, this method is unattainable to determine the crack geometry features such as length, width, shape, and exact location due to block-level implementation.

Another CNN-based crack detector is proposed (Yokoyama & Matsumoto, 2017). This proposed model utilized 2,000 images, categorized into five different classes. The CNN architecture consists of six convolutional layers, three pooling layers, a Leaky ReLU (LReLU) activation, a dropout layer, two fully connected layers, and a SoftMax function for the classification. The evaluation results are not mentioned.

Detection of damage over steel structures using CNN is proposed in (Gulgec et al., 2017). A CNN-based classifier is designed to classify healthy and damaged structures. For the experiment, different sequences of layers and hyperparameters are used to construct 50 networks. The best CNN architecture consists of two convolutional layers, a max pooling, and two fully connected layers. In addition, $tanh$ is used as a nonlinear activation function and SoftMax as a classifier. The evaluation parameters are not shared.

CNN also plays a critical role in pavement crack detection. For the classification of crack images, a PCA-based method is conceived (Xianglong Wang & Hu, 2017). For this research work, iPhone6 is used to capture 60,000 RGB images of 3,264 x 2,448 size. These images are divided into two different sizes, 32 x 32 and 64 x 64, and the transfer learning technique is used for both datasets to detect a crack in this proposed method. The architecture is composed of two convolutional layers, two pooling layers, and two fully connected layers. The $tanh$ function in Equation (2.12) is applied for non-linearity operation.

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.12}$$

The image is first divided into grids and applied CNN for the detection of cracks. Then, the crack grid skeleton is stored, while PCA is applied for crack classification among longitudinal, transverse, and alligator cracks. Precision, recall, and F-measure scores are used for the evaluation. The proposed method achieved 97.2% accuracy for longitudinal crack, 97.6% accuracy for transverse crack, and 90.01% accuracy for alligator crack classification. In addition, the method has achieved a 94.7% F1-Score. However, this method showed a low precision value with the high complexity of the network model.

Crack identification from video frames is time-consuming and need human to inspect. A vision-based crack detection method is developed (F. C. Chen & Jahanshahi, 2018) to detect cracks on the metallic surface. Cracks detection is challenging as they are very thin, tiny, and difficult to detect on the noisy pattern from the metallic surface. The approach used the CNN framework with the naïve Bayes data fusion scheme to analyze video frames. The proposed framework consists of CNN architecture with four convolutional layers followed by batch normalization layer, ELU layer, and max pooling. The last group of layers includes two fully connected layers, an ELU layer, a dropout layer, and a SoftMax classifier. Initially, 5,326 images are collected, which were further converted into 1,47,344 crack patches and 1,49,460 non-crack patches of 120x120 size after rotating

and flipping. The CNN patch detector determines crack patches in each frame. The motion estimator finds the changes in successive frames. Data fusion consists of Spatiotemporal, which registers the crack patches to be determined by Naïve Bayes decision, whether it is a crack or not.

$$\sum_{i=1}^{n}(\log f(s_i^c|C_{crk}) - \log f(s_i^c|C_{ncrk})) = \sum_{i=1}^{n} H_{NB}(s_i^c) > \theta_t \quad (2.13)$$

where $f(s_i^c|C_{crk})$ and $f(s_i^c|C_{ncrk})$ are likelihood functions, $H_{NB}(s_i^c)$ converts detection score to logarithmic likelihood ratio, s_i^c is the score obtained from the CNN for the ith patch, and θ_t is sensitivity controller. The tubeless part generates the bounding box around the cracks. The proposed method achieved 98.3% accuracy, which is higher than other approaches. However, this method also results in a low precision value with the increased complexity of the network model.

Sensors are one of the alternate physical ways to detect structural cracks. But due to complex requirements such as integrating sensors data, installing sensors, temperature variation, and the environmental effect, it is challenging to identify actual cracks. These factors affect the sensing device which leads to generating false output. The problem is overcome by a vision-based crack detecting method proposed by (Y. Cha & Choi, 2017), which uses CNN to process images. The CNN architecture consists of four convolutional layers, two pooling layers, a nonlinear ReLU function, and a SoftMax classification layer, which extracts crack features, and classifies the cracks. The data set consists of 40,000 images, 32,000 are used for training, and 8,000 images are used for validation purposes. The proposed vision-based method achieved 98% accuracy for both training and validation data.

On the other hand, a large dataset with properly labeled images is a surplus for automatic crack detection performance. A data retrieval and annotation method is

proposed by (P. C. Liu & El-Gohary, 2019), which labels crack images collected from the web. These labeled images are further processed for crack classification. A weak CNN classifier is trained on pseudo training data and consists of three convolutional layers, three pooling layers, a flattened layer, two fully connected layers, and an output layer classifier. The classifier's performance was evaluated on recall and precision parameters.

Crack detection from tunnel surfaces has several challenges: poor vision, low texture, large noise, and small datasets for training. These reasons with limited hardware resources can cause difficulty to achieve high accuracy in less operational time. Therefore, a CNN-based method is proposed (Protopapadakis et al., 2019), which only utilizes effective features to produce a better result with less execution time. A CNN classifier annotates the image, eliminates noise through the heuristics mechanism, and finds the image's crack position. Three convolutional layers without a pooling operation are used in the proposed architecture. Accuracy, recall, precision, and F1-Score are calculated to analyze the performance of the model.

2.5.1.2 Classification using CNN Architectures

A robot-based crack inspection and concrete spalling method are proposed in (L. Yang et al., 2017) to avoid human error and cost. The work presented in this research is based on web-based collected images and labeled as actual crack, true spall, or no defect. These images are 278 spalls and 954 cracks images divided into 70% for training, 10% for validation, and 20% for testing. VGG-16 CNN architecture is used to conduct this research. The network is fine-tuned, and the method achieved 93.36% accuracy on the SCCS database and 70% accuracy on the field test for the classification of images.

Furthermore, a comparison among the conventional algorithms (Roberts, Prewitt, Sobel, Laplacian of Gaussian, Butterworth, and Gaussian) and DL-based algorithm (AlexNet DCNN architecture (Krizhevsky & Hinton, 2012)) for crack detection on a

single database is conducted in (Dorafshan et al., 2018a). The dataset contains 1000 images of 2,592 x 4,068 size, divided into 180 sub-images of 256 x 256 which are further split into two categories, 1,574 with cracks and 16,426 without cracks. AlexNet (Krizhevsky & Hinton, 2012) is used as a DL architecture to implement this research. The network is trained in three different modes: fully train, classifier mode, and transfer mode. The transfer learning model shows the significance of DCNN (86% accuracy) compared to conventional edge detectors (53%-79% accuracy) algorithms. However, IPTs and DCNN cannot be comparable as DCNN used sub-images and conventional algorithms used pixels. Furthermore, the IPTs are only designed for a specific dataset, that is why the change in database degrades their performance, and they also need human interaction for a final decision. The DL method, AlexNet is trained for the classification of 1,000 objects which need huge computational time for training.

An end-to-end trainable, automatic crack detection method called DeepCrack is presented in (Zou et al., 2018). The proposed method has encoder-decoder architecture and contains the capability to learn high-level crack features. The model learned the features from four crack datasets, one for training and three for testing. Crack features are learned from hierarchical convolutional operations, where results are fused to capture the line structure. The total loss function is formulated in Equation (2.14).

$$L(W) = \sum_{i=1}^{I} \left(\sum_{k=1}^{K} l\left(F_i^{(k)}; W\right) + l\left(F_i^{fuse}; W\right) \right) \tag{2.14}$$

where l is the number of pixels in an image, $F_i^{(k)}$ is the feature map generated by skip-layer fusion, and F_i^{fuse} multiscale fusion map. The training set has 3,500 images of 512 x 512 size. The method was compared with seven other DL-based methods and one low-level features-based method. The results of DeepCrack show that the proposed method

has better performance in terms of extracting unique high-level features as compared to other methods.

Regarding several crack detection techniques, their performance is structure specific. Furthermore, due to unexpected situations in the real-world scenario, these techniques perform low in the on-site environment. In (B. Kim & Cho, 2018), an automatic vision-based crack detection method using CNN is developed to classify cracks among other crack-like objects such as non-crack, the entire surface, plants, single line, and multiple line joints and edges. Most of the images are collected from the internet and divided into five different classes. The model (AlexNet (Krizhevsky & Hinton, 2012)) is trained through transfer learning over 42,000 images of 227 x 227 resolution to classify crack and non-crack images. The proposed method achieved 96.64% accuracy.

In tunnels, crack detection is a challenging task due to visual issues and lack of continuity. Furthermore, only nighttime is suitable for inspection due to the low traffic. It also requires a skilled person with appropriate precautions. Therefore, a combined approach that uses both DL and traditional image processing tools is proposed to detect, locate and measure the defect (Panella et al., 2018). The dataset consists of 188 crack images. First, image acquisition with three parameters was set up to have a metric for crack detection. The second step includes a low-level algorithm for image processing tasks and elements are segmented. For Image classification, two DL architectures (AlexNet & GoogleNet) were used with transfer learning to compare the accuracy, precision, training, and testing time. The results show that AlexNet performed well on the given dataset as compared to GoogLeNet.

Crack detection on concrete surfaces is challenging due to the lightning effect, surface conditions, and humidity. A DL-based, CNN model for crack detection is developed (Silva & Lucena, 2018) to resolve such issues in an image. A total of 3,500

images are used to perform this experiment. The input image is divided into a patch of 256 x 256 pixels with both cracked and non-cracked images. The training utilized 80% of the image data, while the remaining 20% for testing. VGG-16 architecture parameters are used through transfer learning which achieved 92.27% accuracy for the proposed model.

An image-based road crack detection system is proposed by (Mandal et al., 2018). The method used YOLO v2 (Redmon & Farhadi, 2017) CNN architecture to detect cracks. YOLO v2 looks at the image only once and detects the object using a bounding box with appropriate height and width. The dataset consists of 9,053 images divided into a training set (7,240 images) and a testing set (1,813 images). These images are divided into eight types of cracks. Through transfer learning, weights are tuned from ResNet-101(He et al., 2016a). The YOLO v2 was evaluated on three different metrics; precision, recall, and F1-Score. In addition, the method was compared with a Single-shot Multibox Detector (SSD) (Wei Liu et al., 2016) and a Region-based Convolutional Network (S. Ren et al., 2017). It is also observed that YOLO v2 performed 5% better and faster than SSD and region convolutional neural network (RCNN).

Sewage concrete pipes have been utilized for a long time as their change required both time and money. These pipes can cause leakage, water roots, line breakage, and many other serious threats. Hence, their maintenance must be conducted to avoid serious issues. A crack detection method from sewer pipes using video is proposed (M. Wang & Cheng, 2018). The proposed method utilized a faster region-based CNN (S. Ren et al., 2017). A total of 3,000 CCTV images are captured, 85% images for training and 15% of the images are used for testing. The model achieved 83% of the mean average precision. The result concluded that it is challenging to detect cracks from noisy images.

A GoogLeNet (Szegedy et al., 2015) architecture-based crack detection method is proposed (Zhao & Li, 2018). The architecture was implemented with few modifications in network configurations that operate on 224 x 224 RGB images. A total of 1,250 real-world images were captured using a smartphone. These images are converted into smaller patches of around 60,000. The trained CNN achieved 99.39% accuracy for crack detection.

A dual DL model consisting of CNN and FCN for crack detection is proposed (Liang et al., 2019). The model identifies the crack area and excludes interfering factors in the non-crack area. The FCN model extracts the geometrical crack features such as length and width in the image. The first ten layers of the CNN model consist of convolutional operation, max pooling, batch normalization, and dropout layer, while the subsequent six layers consist of a fully connected layer each with dropout operation. The last SoftMax classification layer classifies the image as crack or non-crack. The convolutional and deconvolutional layers used in the FCN model produce the output of the same input size. This model segments the crack in the classified crack image. The accuracy of the implemented models in this research is 98.60% (CNN) and 99.46% (FCN), respectively.

In (Shengyuan Li & Zhao, 2019), a modified AlexNet (Krizhevsky & Hinton, 2012) CNN architecture is proposed for crack classification. The proposed method integrates with an exhaustive search (sliding window technique) to extract crack and non-crack features. The proposed method overcomes the challenges raised by conventional crack detection methods. The deviation between predicted and actual crack class is calculated through Equation (2.15).

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} f_w\left(x^{(i)} + \lambda r(W)\right) \tag{2.15}$$

where $W$is weight, $f_w\left(x^{(i)}\right)$ is the loss, and $r(W)$ is regularization with weight λ. The dataset has 60,000 images, trained on the proposed network model, and achieved 99.06% validation accuracy and testing accuracy up to 99.09% with a 0.01 training rate.

In (Ahmed et al., 2019), the authors proposed a DL-based road crack detection and damage detection for safe driving. A VGG-16 network is used to detect the crack and a combination of CNN(VGG-16) and RNN(LSTM) is used to classify the crack as a severe or slight crack. The dataset is collected from different sources such as the Concrete Crack image Dataset (Lei Zhang et al., 2016b), SDNET2018 (Dorafshan et al., 2018b), and CrackForest (Shi et al., 2016b). The VGG-16 network has achieved 99.67% crack classification and 97.66% crack type classification.

A hybrid method based on DL and the Bayesian probabilistic approach is proposed by (Fang et al., 2020) for crack detection. The technique uses Faster R-CNN to detect crack patches of suitable scale, further used to train the DCNN regression model for crack orientation recognition. The Bayesian algorithm adopts the location, scaling, and orientation information, which connects local associated detection and suppresses false detection. Faster R-CNN collectively uses two loss functions, i.e., classification and patch box regression losses shown in Equation (2.16).

$$\mathcal{L}(\{\hat{y}_i\},\{\hat{t}_i\}) = \frac{1}{N_{cls}}\sum_i \mathcal{L}_{cls}\left(\hat{y}_i, y_i\right) + \frac{1}{N_{reg}}\sum_i \mathcal{L}_{reg}\left(\hat{t}_i, t_i\right) \tag{2.16}$$

where i is the index of the anchor box, $\hat{y}_i$ is the predicted probability of the anchor being cracked, y_i is the ground-truth label of ith anchor, $\hat{t}_i$ are the predicted coordinates and t_i is the ground truth coordinates of the crack patch box. $\mathcal{L}_{cls}$ is log loss function between crack and non-crack patch, and $\mathcal{L}_{reg}$ is smooth L_1 loss. The DCNN has employed a mean square error loss function for training. This method is evaluated on a new build dataset with 1,675 images with crack and complex backgrounds. The images in this dataset are

collected from various sources such as Google Images, Bing Images, Baidu Images, Crack Forest dataset, and 380 images are captured in a real-world environment. The proposed method is integrated with VGG-16 and ResNet-101 as backbone networks. Similarly, YOLO-v3 and SSD are implemented for comparison, and the proposed method with VGG-16 has achieved the highest precision of 96.8% and F1-Score 92.6%.

A two-step-based crack detection and crack severity classification is proposed (Tran et al., 2020). The authors have used Mask R-CNN for crack detection and identification of severity levels. The severity of the crack is determined through the image processing technique. Cracks are identified based on fatigue, longitudinal and transverse type. Whereas, the severity is divided into three different levels i.e., high, medium, and low. Three other loss functions are used to calculate the multitask loss (classification loss, bounding box loss, and average binary cross-entropy loss). For crack identification, a 5-pixel connectivity image processing technique is selected to retain the crack information. This image is further processed through dilation, erosion, and Gaussian filter to get the final crack. The trained model achieved 92.86% fatigue crack accuracy, 95.15% transverse crack accuracy, and 93.13% longitudinal crack accuracy. However, the image processing technique has gained 87.5% accuracy.

Crack detection in complex pavement conditions is a challenging task due to its complicated structure, nonuniformity, and noise in the crack image. Therefore, a faster R-CNN-based crack detection method (Ibragimov et al., 2020) is proposed, classifying different types of cracks in an image. The faster R-CNN consists of three modules, i.e., Region Proposal Network for proposal generation, CNN for feature extraction, and SVM for object classification. The loss function adopted by Faster R-CNN is shown in Equation (2.17).

$$L(P, u, T^u, v) = L_{cls}(P, u) + [u \geq 1] \sum_{i \in \{x,y,w,h\}} L_{reg}\left(T_i^u, v_i\right) \tag{2.17}$$

where L_{cls} is classification loss, L_{reg} is regression loss, v, and u represent coordinate information and labels. Overall, 2,600 images are used for training and achieved 31.86%, 78.88%, and 87.21% average precision for linear crack, area crack, and patching, respectively.

2.5.1.3 Classification using CNN Architectures with Modules

An automatic image-based crack detection system using CNN is proposed (Chaiyasarn, 2018). A model is proposed to overcome the costly, laborious, and time-consuming crack inspection and detection activity on the historic structure. The proposed model extracts the features from RGB images (a total of 2,934 images of 28 x 28 patches). The implemented CNN architecture consists of two convolutional layers with 32 filters, a non-linear activation function ReLU, a max pooling, a fully connected layer, and a SoftMax layer. Three classifiers (SoftMax by CNN, SVM, and Forest tree) are used with CNN architecture for comparison. The combination of CNN with SVM achieved 74.9% accuracy higher than other classifiers. However, the dataset utilized in this research is not verified, which is one of the most important aspects of DL algorithms that affect the result.

Manual crack detection on a civil structure is inefficient due to cost, the time required to complete the inspection, and skillful assessment. Therefore, a crack identification method based on AlexNet architecture is proposed by (H. Kim et al., 2019), classifying the images as a crack or non-crack. First, the main idea of the crack candidate region (CCR) is imposed in the proposed method that selects the crack candidate from the surface image and represents both crack and non-crack objects. Secondly, the feature obtained from CCR is passed through SURF-based and CNN-based classification. CCR used Sauvola's binarization; for noisy and high-contrast images, shown in Equation (2.18).

$$T = m\left\{1 - k\,.\left(1 - \frac{s}{R}\right)\right\} \tag{2.18}$$

where R is a normalizing factor, k is sensitivity, m is mean, and s is the standard deviation of pixel intensities. The SURF-based classification method consists of feature extraction (interest point detection using Hessian matrix and interest point description using Haar wavelet), visual vocabulary construction (k-means clustering), and classification (SVM). The CNN-based classification consists of five convolutional layers (max pooling and ReLU), three fully connected layers, and SoftMax for classification. The result was evaluated through five performance metrics: precision, recall, F1-Score, accuracy, and computational time. The CNN-based method showed more accurate and efficient results than the SURF-based method for crack identification.

An end-to-end, CNN-based automatic bridge crack detection model is proposed (H. Xu et al., 2019). The network consists of sixteen convolutional layers, three max pooling layers, an atrous spatial pyramid pooling (ASPP) module, and a SoftMax classification. ASPP module contains three Atrous convolutions, used in parallel to obtain larger respective fields without reducing image resolution. The output features from Atrous convolutions are passed to depth-wise separable convolution, which reduces the computational complexity. These multi-scale features contain image context information and are concatenated to increase the network prediction accuracy. The dataset consists of 6,069 images (4,058 crack images and 2,011 background images). The network is trained from scratch using 4,856 images, while 1,213 images are used for testing. The proposed method is evaluated and achieved 96.37% accuracy, 78.11% precision, 100% sensitivity, 95.83% specificity, and 87.71% F1-Score, respectively.

2.5.1.4 Classification using other Networks

CNN is one of the most popular and efficient DL techniques which are gradually getting acceptance in industrial applications. A genetic algorithm (GA), on the other hand, optimizes the parameters accordingly. In this paper (Gibb et al., 2018), a genetic

algorithm is adopted to find the best values of parameters (number of convolutional layers, kernel size, and number of kernels in each layer) to have the optimal structure of CNN for crack detection. The network depth, hyperparameters, and size of layers are evolved by GA in this paper by optimizing these parameters and determining the detailed representation of the image to improve accuracy. The dataset has 3,000 images for training and 600 for testing. A fitness function based on correctly classified images over total classified images measures the accuracy. The genetic algorithm uses (μ, λ) population size and random generation of children. Therefore, the individual with higher fitness has a higher probability of carrying the next generation. The optimized parameters obtained after applying GA are eleven convolutional layers with twenty kernels of 15 x 15 size.

A one-stage crack detector, CF-Net is proposed by (Xia et al., 2020) to detect real-time cracks on railway sleepers. The proposed network used a CF module along with a modified loss function. The proposed network is based on RetinaNet, which uses ResNet-50 and Feature Pyramid Network for feature extraction. The two networks are used to predict the crack region and coordinates of the crack region. The classification network adopted focal loss, whereas the regression network adopted the smooth-flat loss function shown in Equation (2.19),

$$smooth_{flat}(x) = \begin{cases} 0.25x^4 & if\ |x| < 1 \\ |x| - 0.75, & othersise. \end{cases} \quad (2.19)$$

where x is the coordinate deviation between ground truth and predicted box. The network used 54,095 self-collected images from railway maintenance work to train the proposed network and achieved 98.1% accuracy, 92.10% precision, 79.4% recall, and 85.20% F1-Score, respectively.

2.5.1.5 Interpretation of Crack Detection Through Classification

The crack classification methods can only classify a patch or whole image as a crack region or a non-crack region. The classification methods used the sliding window technique, which scans the border pixels of the image only once and scans multiple times other pixels. If the crack locates at the corner/edge of an image, the information will not be properly manipulated because the pooling layer on CNN downsized the original data and lost the precious information. The other drawback that degrades the performance of the sliding window method is that thousands of image patches are processed patch by patch. The sliding window needs to apply CNN on multiple image patches and at numerous locations with different filters, which involves a lot of computation. However, only a few computations are enough to extract valuable crack features as these shared features are not used between overlapping small patches. This method is computationally inefficient as it causes an increase in the computation expense. Furthermore, the sliding window does not provide the exact location of the crack in the image because localization depends on the sliding window size. Therefore, crack detection through a sliding window technique for crack feature extraction is somehow incapable of crack localization and could not achieve pixel-level accuracies.

The summary of the related work discussed in this section is illustrated in Table 2.1.

Table 2.1: Related work summary for crack classification

References	Architecture	Dataset	Loss function
(Lei Zhang et al., 2016a)	DCNN	Self-Collected Images	CE
(Pauly et al., 2017)	DCNN	Self-Collected Images	CE
(Y. J. Cha et al., 2017)	DCNN	Self-Collected Images	CE
(Yokoyama & Matsumoto, 2017)	DCNN	Self-Collected Images	CE
(Gulgec et al., 2017)	DCNN	Self-Collected Images	Negative Log-likelihood

References	Architecture	Dataset	Loss function
(Xianglong Wang & Hu, 2017)	DCNN	Self-Collected Images	CE
(F. C. Chen & Jahanshahi, 2018)	DCNN	Self-Collected Images	CE
(Y. Cha & Choi, 2017)	DCNN	Self-Collected Images	SoftMax
(P. C. Liu & El-Gohary, 2019)	DCNN	Collected from the Internet	CE
(Protopapadakis et al., 2019)	DCNN	Self-Collected Images	CE
(L. Yang et al., 2017)	CNN (VGG16)	Concrete Structure Spalling and Crack	CE
(Dorafshan et al., 2018a)	CNN (AlexNet)	Self-Collected Images	CE
(Zou et al., 2018)	CNN (VGG16)	CrackTree260 CrackLS315 Stone331 CRKWH100	WCE
(B. Kim & Cho, 2018)	CNN (AlexNet)	Collected from the Internet	CE
(Panella et al., 2018)	CNN (AlexNet and GoogLeNet)	Self-Collected Images	CE
(Silva & Lucena, 2018)	CNN (VGG16)	Self-Collected Images	CE
(Mandal et al., 2018)	YOLO-v2	Microsoft COCO + Self-Collected Images	Sum-Squared Error Loss Function
(M. Wang & Cheng, 2018)	Faster R-CNN	Self-Collected Images	Log Loss Function
(Zhao & Li, 2018)	CNN (GoogLeNet)	Self-Collected Images	CE
(Liang et al., 2019)	CNN & FCN	Self-Collected Images	CE
(Shengyuan Li & Zhao, 2019)	CNN (AlexNet)	Self-Collected Images	CE
(Ahmed et al., 2019)	CNN (VGG-16)	SDNET2018 CrackForest CCDataset	CE
(Fang et al., 2020)	Faster R-CNN	Self-Collected Images	Log Loss, Regression Loss, and MSE Loss
(Tran et al., 2020)	Mask R-CNN	Self-Collected Images	Classification Loss, Bounding Box Loss, and CE Loss
(Ibragimov et al., 2020)	Faster R-CNN	Self-Collected Images	Regression Loss and CE

References	Architecture	Dataset	Loss function
(Chaiyasarn, 2018)	CNN + SVM	Self-Collected Images	Hinge Loss Function
(H. Kim et al., 2019)	CNN + SURF	Self-Collected Images	CE
(H. Xu et al., 2019)	CNN (ASPP module)	Self-Collected Images	CE
(Gibb et al., 2018)	CNN (Genetic Algorithm)	Self-Collected Images	CE
(Xia et al., 2020)	CF-Net	Self-Collected Images	Focal Loss and Smooth-Flat

2.5.2 Crack Segmentation Techniques

Segmentation is an alternative method for crack detection. It is a pixel-level-based technique used for CV tasks. It generates label masks as an output to segment crack and non-crack pixels. It determines the crack geometry and locates the cracks by producing the segmented crack image. In recent years, CNN has achieved great success in the research area of crack pixel segmentation.

2.5.2.1 Segmentation using DCNN

The complex pavement conditions make crack detection a challenging task in a real-time scenario. A DL-based method (Fan et al., 2018a) for a multi-label classification problem is proposed to find cracks under complex pavement conditions. The proposed architecture learns the crack structure from raw images of two different datasets for the experiment, (i) CFD with 118 (72 for training and 46 for testing) RGB images of size 324 x 480 and (ii) AigleRN with 38 (24 for training and 14 for testing) grayscale images. The input images are converted into small patches of 27 x 27 x 3 for RGB and 27 x 27 x 1 for grayscale images. To balance the ratio between crack pixels and non-crack pixels in a crack image, a ratio of 1:3 is set as non-crack pixels have more representation than crack pixels in the crack image. These patches pass through the CNN architecture for training. Cracks are manually labeled, and the model is trained over them to predict the neighboring crack pixels. CNN architecture consists of four convolutional layers, two max pooling layers, and three fully connected layers. The method is tested on two public

datasets and compared the results with four other existing methods. The proposed method is evaluated on precision, recall, and F1-Score with CFD and AigleRN datasets and outperforms other methods.

A deep FCN method for semantic segmentation on concrete crack images is proposed for crack detection and density evaluation (Vu & Duc, 2019). In this research, pixel classification is done to monitor and inspect the civil infrastructure. Three different pre-trained network architectures VGG Net (Vo et al., 2018), ResNet (He et al., 2016a), and InceptionV3 (Szegedy et al., 2016) on the concrete crack dataset, are evaluated for the implementation of the proposed method. FCN (J. Long et al., 2015) and VGG16 (Vo et al., 2018) performed better than the other two architectures. The encoder-decoder FCN architecture is trained over 40,000 images and achieved 90% accuracy for end-to-end pixel-based classification and segmentation.

An improved deep fully CNN named CrackSegNet is proposed by (Y. Ren et al., 2020) for pixel-wise crack segmentation in tunnels. The proposed network consists of an encoder, decoder, dilated convolutions, spatial pyramid max pooling, and skip connections. The backbone network of the encoder path is a modified version of VGG-16 architecture. A dataset of 409 crack images that has 4032 x 3016-pixel resolution is captured. These images are cropped into 512 x 512 resolution, and 736 cropped images are used for training and 183 for the test. The focal loss function is used to address the imbalanced class problem. The proposed network is evaluated with different layers and methods and achieved 99% accuracy in almost every case.

2.5.2.2 Segmentation using U-Net

A U-Net-based pixel-level crack classification method is proposed by (David Jenkins et al., 2018). An automatic crack detection method is designed to segment the crack from noise, illumination, and textural surface in this research. U-Net architecture is

implemented through a patch-based training methodology for image segmentation. The limited dataset and a significant reduction in resolution by down-sampling are not required for pixel-level classification. Thus, the method is developed to overcome the problem of losing fine details of the image during the down-sampling (encoder) process faced by SegNet (Badrinarayanan et al., 2017). The U-Net architecture consists of an encoder for down-sampling and a decoder for up-sampling. The output of each encoder layer is concatenated with the decoder layer to save the fine details. The encoder consists of two convolutional layers, two ReLU, and a max pooling layer. The decoder consists of two convolutional layers, two ReLU, and an up-sampling layer followed by a SoftMax layer for classification. The dataset has 118 grayscale crack images (80 for training, 20 for validation, and 18 for testing). Test image patches 48 x 48 size is processed using the sliding window method. CrackForest dataset is used on precision, F1-Score, and recall metrics for the evaluation. The result outperformed the state-of-the-art methods on two out of three evaluation metrics.

Another U-Net-based end-to-end semantic segmentation network is proposed for pixel-level crack classification (Ji et al., 2018). In this research, the localization issue of sliding window-based methods is addressed by using pixel-level detection. U-Net architecture with some modifications (input image size, zero paddings in convolutional layers against shrinking, Adam optimizer for faster convergence, and drop-out layers to avoid overfitting) is used. The dataset contains 30 raw images and an annotated mask (JSON file) where 1 is labeled as a crack and 0 as a non-crack pixel. These raw images are further divided into 512 x 512 sizes (200 images): 70% of the images are used for training, 15% for validation, and 15% for testing purposes. The binary cross-entropy loss function is used for the pixel-level classification. The evaluation metrics over which this method is evaluated are precision, recall, and F1-Score. These methods are compared to

the Canny edge detector and Sobel edge detection method. The proposed method achieved high accuracy and had a robust performance for crack detection.

A U-Net-based automatic pixel-level crack segmentation method is proposed (Cheng et al., 2018). The proposed method resolved the imbalanced data problem as a crack image contains many non-crack pixels compared to crack pixels. The encoder-decoder CNN architecture detects and segments the cracks. This method processes the entire image and generates segments directly without dividing the image into small patches. In addition, a new lost function that finds weight according to minimal distance is introduced to achieve better accuracy.

$$L(x) = -\sum_{i=1}^{K} (d_i^I(x)\beta_i y_i(x) \log \hat{y}_i(x) + d_i^O(x)(1-\beta_i)(1 - y_i(x)) \log(1-\hat{y}_i(x))) \tag{2.20}$$

where β_i is the class sample ratio, d_i^I is inner and d_i^O is outer Euclidean distance transform map. The architecture takes a grayscale image as an input and produces a segmented output image of the same size by classifying between a crack and a non-crack pixel. The input images do not require pre-processing like creating image patches. Furthermore, to resolve the imbalanced data problem number, a balanced cross-entropy loss function based on chamfer distance is used to predict the probability of each pixel.

A U-Net-based concrete crack detection method is used (Z. Liu et al., 2019), which utilized the focal loss function for updating weights through the Adam optimization method. The training process utilized 57 images while 27 images are used for testing. k-fold cross-validation is used to evaluate the ML model with k=3. The proposed method achieved 90% precision, 91% recall, and 90% F1-Score, respectively.

Manual crack image annotation is a challenging and lengthy task that requires a trained expert to label the cracks. A U-Net-based, fully CNN is proposed (Konig et al., 2019), which uses a small dataset to train the network. The U-Net architecture is modified by adding residual blocks and attention gates to crack image segmentation. The encoder block consists of a 1 x 1 linear transformation filter followed by four convolutional blocks and a ReLU activation function. The output of the 1 x 1 linear transformation filter is added to the output of the last activation function. On the other hand, the attention gate is used to retain the spatial relevant features and generates a scaling coefficient a_i^l, which is added in decoder block through skip connections.

$$a_i^l = \sigma_2(q_{att}^l(x_i^l, u_i^l)) \tag{2.21}$$

$$q_{att}^l(x_i^l, u_i^l) = W_v(\sigma_1(W_x x_i^l + W_u u_i^l + b_u) + b_v) \tag{2.22}$$

where W_v, W_u, and W_x are 1 x 1 linear transformation filters with biases b_u and b_v. ReLU activation function is presented as σ_1 and Sigmoid as σ_2. The term x_i^l are the feature of lth U-Net architecture level and ith pixel of feature map x and u_i^l is the previous up-sampling operation. The network uses two loss functions, i.e., cross-entropy loss and dice loss function, to minimize the difference between ground truth and predicted mask. The encoder-decoder U-Net architecture is trained on the CFD and AigleRN datasets and achieved 94.94% and 89.86% F1-Score, respectively.

In (Lau et al., 2020), a U-Net-based CNN architecture is designed to segment the crack images with an irregular pattern, lighting conditions, and noisy images. The encoder part of U-Net is replaced with ResNet-34. The modified network is integrated with spatial and channel squeeze excitation (SCSE) modules that improve the result by a small margin in precision and F1-Score. The dice coefficient loss function is used to measure the difference between predicted and ground-truth masks. The network is evaluated on two

datasets and achieved a 96% F1-Score for the Crack500 dataset and a 73% F1-Score on the CFD dataset.

A pixel-level crack detection model is proposed by (Lingxin Zhang et al., 2020), with U-Net architecture as a base network. In this research, the effect of the dataset and network depth on training time, accuracy, and speed are analyzed. The original U-Net is modified, and four new CrackU-Net architectures are proposed based on different convolutional layers. CrackU-Net7, CrackU-Net11, CrackU-Net15, and CrackU-Net19 have seven, eleven, fifteen, and nineteen convolutional layers. Furthermore, these networks have different network levels in the symmetric structure. The generalized dice loss function is adopted to resolve the imbalanced class issue. The dataset has 1,200 augmented images collected from the Internet, Lab, and other different literature. The first experiment is performed to analyze the effect through four different datasets (100, 200, 400, and 800). The CrackU-Net19 performed best in these datasets, other than the overfitting problem in smaller sets. The second experiment is performed to analyze the effect of network depth, where CrackU-Net7 achieved 97.75%, CrackU-Net11 achieved 99.31%, CrackU-Net15 achieved 99.44, and CrackU-Net19 achieved 99.36% accuracy. The training time of the proposed CrackU-Net (7, 11, 15, and 19) model was 1.1, 1.4, 1.7, and 1.9 hours, respectively.

The dataset of crack images is usually captured by a smartphone that does not possess standard quality resolution. To accurately extract the crack information from these images, a DCNN based U-Net architecture is proposed by (Huyan et al., 2020) for pixel-wise crack detection. The proposed CrackU-Net architecture is modified such that mean pooling operation and fixed convolutional size are used. In addition, the cross-entropy loss function is used as a loss function. The dataset has 3,000 crack images captured from a high-speed action camera and smartphone, 2,400 images are used for training, and 600

are used for testing. The proposed network reached 99.01% accuracy, 98.56% precision, 97.98% recall, and 98.42% F1-Score. Thus, CrackU-Net significantly outperforms the FCN and U-Net architecture.

Labeling crack pixels for training is a manual and time-consuming task. Therefore, a patch-based weakly supervised semantic segmentation network (Dong et al., 2020) for crack detection is proposed which uses image level annotation through crack topology. Furthermore, a localization method is developed to extract locational information from the crack. The technique initially crops the image into two classes, i.e., true; if it contains any part of the crack and false; if it does not contain any crack portion. The cropped images are further passed through the localization method to extract the activation crack mapping. The DenseCRF in the next step generates labels used with initially cropped images to train the network. The predicted output joint with a fusion strategy to obtain the final output. U-Net with a ResNet architecture is used as a backbone network along with balanced weighted cross-entropy loss. A position attention module is used in ResNet to involve the contextual information of the feature map effectively. Two crack datasets called DeepCrack (300 images), and Cls-dataset (20,000 images) are used for training and 237 images are used for testing.

A two-step pavement crack detection and segmentation method (J. Liu et al., 2020) is proposed to improve crack detection and segmentation accuracy. The initial step used modified YOLOv3 to detect cracks in images and the last step uses modified U-Net to segment the detected cracks. First, the encoder part of U-Net is replaced with ResNet-34. Then, a spatial channel squeeze and excitation module are integrated with the up-sampling layer, detecting and segmenting the crack. Self-captured crack images along with the CFD dataset are used to train the network. Out of 27,966 images, 16,780 images are used for training, and 11186 are used for testing. Dice coefficient loss is used for error

calculation. The proposed method performed better than other methods by achieving 97.24% precision, 94.31% recall, and 95.75% F1-Score, respectively.

2.5.2.3 Segmentation using VGG

Evaluation of building cracks is time-consuming and required skilled laborers to perform crack identification tasks. Cracks are classified into various categories such as orientation, width, length, and type. Therefore, its description, detection, and storage are costly and challenging to implement. A computer vision-based, FCN technique for crack detection is proposed to detect cracks automatically at the pixel level (X. Yang et al., 2018). FCN is an end-to-end convolutional network with a down-sampling part (convolutional, pooling, and dropout layer) that uses VGG-19 and an up-sampling part (deconvolutional layer) for dense prediction. The information from up and down sampling is concatenated to determine the type and location of the crack. The architecture consists of a convolutional layer, ReLU, max pooling, deconvolutional layer, and SoftMax function. The model adopted the exponential staircase decay function defined by Equation (2.23) to reduce the initial learning rate.

$$lr_t = lr_0 . r_d \left\lfloor \frac{t}{t_{max}} \right\rfloor \tag{2.23}$$

where lr is learning rate, r_d is the decay rate, and t is the iteration step. 800 images (80% for training and 20% for testing) are collected for the experiment and are resized to 224 x 224. Cross entropy is used as a loss function and evaluated on precision, recall, and F1-Score. The FCN's feature maps produce low prediction and poor resolution due to information lost during down-sampling.

CrackNet-V based on a deep neural network is proposed (Fei et al., 2019) for pixel-level crack detection. The method determines a specific pixel in a particular region on a 3D asphalt pavement image and classifies that pixel as a crack or non-crack. The CrackNet-V is faster than CrackNet (A. Zhang et al., 2017) due to fewer parameters

(63,113 parameters) and preserves the information of corner pixels because it excludes the pooling layer. CrackNet-V is based on VGG architecture with ten layers. The network has a pre-processing layer (for rectification of unevenness due to the cross slope, rutting, and deformation), eight convolutional layers followed by Leaky Rectified Tanh shown in Equation (2.24) (which provides high accuracy in detecting narrow and fine cracks) and a Sigmoid-like function in the final layer.

$$\sigma(x) = \max\left(1.7159\, tanh\ (\frac{2}{3}x),0\right) + \min(0.1x, 0) \tag{2.24}$$

For error minimization, a cross-entropy loss function is used that differentiates between the predicted and actual pixel.

$$\mathrm{C} = \frac{3}{4}[(t - 1.7159)ln(1.7159 - a) - (t - 1.7159)ln(1.7159 + a)] + const + L_2\ Item \tag{2.25}$$

where t is the targeted pixel, a is the predicted pixel, const (3.174) is used to avoid negative error, and L_2 Item is a weighted decay parameter. The database consists of 3D pavement images and ground truth images. Average pooling operation is applied to reduce the original image size from 4096 x 2048 to 512 x 256. The network was trained with 2,568 images, 15 for validation and 500 for testing. The architecture is evaluated on Precision, Recall, and F1 score and has a high value of Recall and F1-Score but slightly low precision. The CrackNet-V was also compared with CrackForest (SVM) (Shi et al., 2016b) and achieved higher Precision and F1-Score. The CrackNet-V is also evaluated on two different activation functions Leaky Rectifier Tanh and Leaky ReLU. The results show that with Leaky Rectifier Tanh, CrackNet-V can detect more fine and shallow cracks.

An end-to-end pixel-wise crack segmentation method is proposed by (Y. Liu et al., 2019), which used deep hierarchal CNN. The architecture adopted a VGG-16 network with N output classes in the SoftMax layer, which labels the predicted values to each

pixel. The architecture excludes a fully connected layer because of computation, time/memory consumption, and shrinks output which is not useful to generate the result. After each convolutional layer, a side-output layer with deep supervision is inserted that has different scales. These side layers contain multi-scale and multi-level features which are concatenated to form the final output. Each side layer contributes to generating the output loss. The overall output loss is calculated using Equation (2.26).

$$\mathrm{L} = L_{side}(I, G, W, w) + L_{fuse}(I, G, W) \tag{2.26}$$

where L_{side} is an image-level loss for side output, L_{fuse} is concatenated final fused loss, I is an input image, G is the total number of pixels, W is all network parameters, and w is class weights. The output from the fused result is passed to Deeply Supervised Nets (DSN) which provides direct supervision for features. Furthermore, FCN learns and aggregates multi-scale and multi-level features with DSN. Guided filtering and conditional random fields are used to refine the prediction. The results demonstrate that the method has achieved comparable results with state-of-the-art methods. It is also concluded that lower CNN layers have better information on the crack region but are unhelpful due to local noise, while deeper layers have anti-noise ability but have poor performance in boundary segmentation. The results obtained from end-to-end pixel-wise crack segmentation are better than predicting bounding boxes.

Traditional crack detection methods possess low accuracy and generalization. A novel context-aware pixel-wise deep semantic segmentation network is proposed by (X. Zhang et al., 2019) to effectively detect and segment cracks from an image. The framework can support any arbitrary size image (both RGB and grayscale) as input. The detection system has three main parts and five major components; the first part is the localization of image patches (using global non-overlapping (a) sliding windows which includes possible crack areas and (b) Sobel-edge detector (NMIPS) for extracting textual information), the second part is pixel wise crack prediction ((c) SegNet is used to assign

predictions to crack pixels, the encoder has a convolutional layer, BN, ReLU, and non-overlapping max pooling while decoder has convolutional layers, BN, ReLU and memorized max pooling) and the third part is crosspatch contextual information contains cross-state potential function shown in Equation (2.27) and the cross-space potential function shown in Equation (2.28) ((d) CAOPF is used to fuse the prediction patches (e) MSMP to reduce computation and processing time while maintaining the performance).

$$x\left(S_{i,j}^{k}, S_{i,j}^{l}\right) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma_s^2}} e^{-\frac{\left(|S_{i,j}^{k} - S_{i,j}^{l}| - \mu_s\right)^2}{2\sigma_s^2}}, & ||C^k - C^l|| \le \alpha \\ 1, & otherwise \end{cases} \tag{2.27}$$

where $x\left(S_{i,j}^{k}, S_{i,j}^{l}\right)$ is the cross-state potential function, $S_{i,j}^{k}$ is the binary pixel states of overlapping image patch k at position i,j, $S_{i,j}^{l}$ is the binary pixel states of overlapping image patch l at position i,j, $\mu_s$and σ_s^2 is mean and variance of Gaussian distribution between $S_{i,j}^{k}$ and $S_{i,j}^{l}$, C^k and $C^l$is the Euclidean distance between the central pixel coordinate of image patch k and l.

$$\varphi(C^k, C^l) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(\frac{1}{\rho}||C^k - C^l|| - \mu_c)^2}{2\sigma_c^2}}, & ||C^k - C^l|| \le \alpha \\ 1, & otherwise \end{cases} \tag{2.28}$$

where $\varphi(C^k, C^l)$ is the cross-space potential function between C^k and C^l, ρ is the normalization factor, μ_c is mean and σ_c^2 is a variance of Gaussian distribution between C^k and C^l. The network is evaluated on F1-Score, Precision, and Recall metrics on three different datasets CrackForest Dataset (CFD) and Tomorrows Road Infrastructure Monitoring, Management Dataset (TRIMMD), and a Customized Field Test Dataset (CFTD).

A unified neural network-based VGG-19 network and CRF, called DilaSeg-CRF is proposed (M. Wang & Cheng, 2019). The network segments the cracks from the sewer pipes. The simple DilaSeg model uses the RNN layer instead of dense CRF and addresses

spatial information loss. It uses dilated convolution, enlarging the respective field, and includes more contextual information. In addition, the dense conditional random field is used to improve segmentation. The other network components are parallel multiscale layers to maintain the aspect ratio of certain objects, bilinear interpolation to up samples the integrated pixels of parallel dilated convolution, and the SoftMax layer is replaced with RNN layers. The proposed model achieved Pixel Accuracy 98.69%, mean Pixel Accuracy 91.57%, mean Intersection over Union (IoU) 84.85%, and frequency weighted IoU 97.47% to evaluate segmentation accuracy. To train the network, 1,880 CCTV images of 512 x 256 resolution are used. This study also considered the computational cost.

A DL-based crack detection method is proposed by (J. S. Lee et al., 2020), which measures the maximum crack width. The CSN is based on VGG-16. The architecture includes a shape-sensitive kernel called CK, a moving kernel within an image that overlaps with a crack and creates a greater chance of crack pixel detection. Three CK models are proposed in this paper that differ by the activation function, the number of convolutional layers, and the orientation of kernels. A total of 2750 images, which subsequently divided into a ratio of 7:1:2 for training, validation, and testing. The proposed network with CK module 2 achieved 97% crack accuracy and 99% non-crack accuracy, respectively.

Based on SegNet architecture, a fully CNN called pavement and bridge crack segmentation network (PCSN) is proposed by (T. Chen et al., 2020) for crack pixel segmentation. The encoder portion of architecture consists of VGG-16. The proposed PCSN architecture has eleven blocks with five convolutional blocks at the encoder side, where each block has 2 or 3 convolutional layers along with max pooling. The decoder part consists of five blocks, each with an up-sampling layer and 2 or 3 convolutional

layers. The last SoftMax layer transforms the feature maps into class probabilities of each pixel. The network is pre-trained with VGG-16 weighting parameters for initialization. The training process accepts RGB images of size 512 x 512. The dataset contains 10,000 images captured from a high-resolution camera; randomly, 5,000 images are selected for training and 2,500 for validation and testing. The proposed network adopted a cross-entropy loss function for error calculation and Adadelta as a network optimizer. As a result, the PCSN achieved 83% mean pixel accuracy.

A ResNeXt based framework is proposed (G. Li et al., 2020) to detect cracks on bridge concrete structures. The proposed network is a modified version of the original ResNeXt, combining the VGG architecture and Inception network. The dataset is built by capturing 25,358 crack images using a DSLR camera along with a distance meter, flash, telephoto lens, and various sensors. These captured crack images are divided into five different types of images based on shape and orientation (complex cracks, wide cracks, oblique cracks, intersecting cracks, and reticulated cracks). Furthermore, the rotation augmentation method is used to increase the dataset size up to 45,358 images. The dataset is divided into 36,286 training, 4,536 validation, and 4,536 test images, respectively. The proposed method achieved 95.73% accuracy, 91.7% precision, 93.1% recall, and 92.5% F1-Score, respectively. The framework is concatenated with a post-processing method and includes the binarization of adaptive crack width that determines crack pixels in the images.

A weakly-supervised crack detection method is proposed (L. Xu et al., 2020), which uses fewer training images and labels to identify the surface crack. The network model consists of shared layers (VGG architecture) that extract feature information from the input image. LNet is used, which has four convolutional layers to generate heat maps to predict defect location. The DNet uses the residual spatial attention module (RSAM) to

produce the defect probability through the Sigmoid activation function. The images are classified as defected or non-defected images using the binary cross-entropy loss function. If the image is classified as defective, the pixels are divided into two categories. The KolektorSDD dataset is used to train the network with a novel sampling process to resolve the imbalanced dataset issue achieved 99.5% recognition accuracy.

2.5.2.4 Segmentation using CNN Architectures

In (Dorafshan et al., 2018b), an image dataset of 56,000 cracks and no crack images are built. These images are used to fully train AlexNet DCNN. The author used a 16-MP Nikon camera with a 4,068 x 3,456 image resolution. Each image is then segmented into 256 x 256 sub-images.

A CNN-based architecture called CrackNet is proposed by (A. Zhang et al., 2017) to automatically detect the cracks at the pixel level. CrackNet is used for pavement crack detection on 3D asphalt surfaces with pixel-perfect accuracy. It has a slightly different architecture as compared to general CNN. It excludes the pooling layer which decreases the number of features. It consists of two convolutional layers (the feature detector size of a second convolutional layer is 1 x 1), two fully connected layers, a nonlinear ReLU function, a dropout layer to avoid overfitting, and a Sigmoid layer. The feature maps are extracted before the information is passed in architecture using 360-line filters which are obtained through multiple orientational filters. A total of 2,000 3D pavement images of size 512 x 1042 (1,800 for training and 200 for testing) are used. The learning process of CrackNet includes mini-batch gradient descent, cross-entropy, dropout, momentum, and normalized initialization. The results are evaluated and compared with traditional ML and imaging algorithms on Precision, Recall, and F1-Score. Although CrackNet performed better than other methods, it requires substantial processing time. The hairline cracks are difficult to detect through this because it lacks appropriate data for training. CrackNet

utilized the feature generator which has no learnable parameters resulting in weak learning capability. The use of down-sampling may also cause some information to lose which leads to the wrong prediction. The method that extracted the features before applying them to the network cannot extract lines of small width.

Furthermore, CrackNet-II (A. Zhang et al., 2018) is proposed, an improved version of CrackNet (A. Zhang et al., 2017). It is a DL-based pavement crack detection method for faster performance. The detection is performed at the pixel level which does not affect the width and height of the image by varying layer size. A multichannel 1 x 1 convolutional layer is used to learn the difference between the background pixel and crack pixel. Local-level pixel comparison is performed at the convolutional layer. The two major modifications in CrackeNet-II are the improvement in learning capability by eliminating feature generators and deeper architecture with more hidden layers but fewer parameters. The CrackNet-II has 10 convolutional layers (general-purpose and 1 x 1 convolutional layers). The training includes normalized initialization, batch normalization, dropout, mini-batch gradient descent, cross-entropy, and momentum. The dataset consists of 3,000 3D pavement images of 1024 x 512 in size with manual preparation of ground truth images of crack. The database is divided into 2500 training images, 300 validation images and 200 testing images, respectively. The performance of CrackNet II over CrackNet is five times faster and has achieved 90.20% in precision, 89.06% in recall and 89.62% F1-Score. The major achievement of CrackNet II is detecting hairline cracks by avoiding noises. CrackNet-II does not acquire a global context, which is used to differentiate between noise patterns and pavement cracks. It also misclassifies shoulder drop-off patterns and pavement edge as a longitudinal crack. On concrete surfaces, pavement joints and grooves are identified as false-positive errors. Individual crack is detected as an unconnected part and not positive every time about the continuity of each detected crack.

A dense-dilation fully- CNN is proposed in (Kaige Zhang et al., 2018) for high-resolution images. The method detects cracks from any input image size by inserting the full image only once. The image classification network is trained on patch images (crack, sealed crack, and background) of 227 x 227 size for the implementation. This way, cracks can be localized with better accuracy. A dense dilation network transfers the low-level and middle-level features to classification networks. AlexNet (Krizhevsky & Hinton, 2012) is implemented as a source network with a few alterations in the original network for training. The result was evaluated using precision, recall, and HD-score. These results are compared with CrackIT (Oliveira & Correia, 2014), CrackForest (Shi et al., 2016b), and (Dorafshan et al., 2018a). The result of the proposed method showed good accuracy in locating cracks and can differentiate cracks with sealed cracks and background efficiently.

Cracks are an alarming sign for civil structures. For the inspection of cracks regularly, many image processing and machine learning-based crack detection algorithms have been developed. A pixel-level crack delineation network (CDN) is proposed (Ni et al., 2019) for quick and accurate detection and segmentation of cracks at the pixel level. The framework consists of GoogLeNet architecture and CDN. GoogLeNet classifies the cracks at pixel level by extracting features using three network components (i) convolution pooling layer (reduces data dimensions, expands pattern linkage, and achieves transformation), (ii) inception (increases the depth and width of NN and decreases parameters), and (iii) dropout (regularization technique to prevent overfitting). The CDN has two network components, convolutional feature-map fusion, and consecutive convolutional layers. For updating weight, an SGD optimizer is used along with a cross-entropy as a cost function to update large errors fast and small errors slow. The dataset has 800 (640 training and 160 testing images) raw images of size 4,000 x 6,000, further divided into 224 x 224 patches. These patches are discriminated into the

classification and delineation groups. The evaluation is done from Net-1 to Net-5 (lower to higher) and Net-6 to Net-9 (higher to lower). Thirteen networks are used for different feature map fusion by extracting and integrating the previous order with current order features. Precision, recall, and F-measure is used to record performance. Different orders are also noted for different types of cracks such as simple cracks, hairline cracks, cracks with complex texture in the background, and cracks with artificial marks. All networks performed well except Net-6 because it uses higher-order feature maps.

Crack localization contains useful information for crack detection methods for locating the crack correctly in the image. Image-based crack detection methods can identify the location and severity of the crack. However, images that contain objects other than cracks are difficult to process. In (Bang et al., 2019), a pixel-level crack detection method is proposed using the CNN encoder for feature extraction and decoder for localizing the cracks. The model detects and classifies the pixels as crack or non-crack pixels. The dataset consists of 527 black-box images of size 1,920 x 1,080 x 3 (427 for training and 100 for testing). Black-box images have relative quality and complex background as it includes every visible object in the image such as traffic signals, trees, house, building, etc. The proposed network uses Resnet-152 for encoding and deconvolution for decoding (SegNet, FCN, and ZFNet). Cross-entropy is used as a cost function with the Adam optimizer. A comparative result of the proposed method with some other architectures VGG-16, ResNet-50, ResNet-101, ResNet-200, and ResNet-152 without transfer learning while the rest of the networks adopt transfer learning. The network is evaluated on parameters such as recall, precision, F1-Score, and IoU. The ResNet-152 achieved the best results as compared to other networks. Although the proposed method performed well using ResNet-152, it could not help to detect fine cracks because of a small dataset, which does not train the network well.

A pixel-wise crack detection network is proposed (D. Lee et al., 2019). The proposed method is based on end-to-end DL for image segmentation called the Crack Segmentation Network. The network is specially designed for complex crack patterns. The CSN architecture scans the whole image, it has five convolutional blocks and one recovery block. Each convolutional block has a convolutional layer, pooling layers, ReLU/ELU, and batch normalization. The recovery block is called the transpose layer which performs pixel-wise crack prediction and SoftMax performs pixel-wise classification. A Brownian motion process is used to imitate crack shape, presented in Equation (2.29)

$$\beta_k = \sum_{i=1}^{k} v_i \tag{2.29}$$

where β_k is Brownian motion and v_i independent random Gaussian increment. The CSN is trained over the MS-COCO database with RGB images of size 720 x 1280 and evaluated the architecture through accuracy, precision, and recall.

2.5.2.5 Segmentation using R-CNN and its Variants

The accuracy of crack detection methods is influenced due to many factors that exist in the crack images such as traces of tie-rod holes, and formworks. A crack detection method based on DL semantic segmentation (Yamane & Chun, 2020) is used to overcome the shadow and dirt conditions on crack images. Mask R-CNN is used to train the 45 self-captured images which are subdivided into 24,336 training images and 6,048 testing images through the K-fold validation technique. The method also used traces of tie-rod holes and formworks to exclude the false detected area and increase the crack detection accuracy. The proposed method achieved 99.15% accuracy, 48.62% F1-Score, recall 78.81%, and 39.24% precision before excluding the false detection and 99.21 % accuracy, 49.94% F1-Score, recall 78.47, and 40.44% precision after excluding the false detection.

Cracks with complex backgrounds are difficult to detect because they contain crack-like features. A crack detection, localization, and quantification method are proposed in

(Kang et al., 2020), which combined Faster R-CNN based on ResNet-50 for crack detection, a modified TuFF that uses CLAHE for crack pixel segmentation, and DTM measures the crack orientation. The dataset has 1,200 training images and 100 validation images selected from the publicly available dataset. The 100 images, selected for testing the trained network, contain thin, medium, and wide cracks, captured near the University of Manitoba, the Bridgewater Forest area in Winnipeg, along with other images collected from the Internet. The Faster R-CNN had 95% average precision, modified TuFF scored 83% IoU, and DTM achieves 93% accuracy.

Automatic crack detection is a challenging task, especially in the complex background and complicated shapes. An instance segmentation network is proposed by (Y. Zhang et al., 2020) for pavement crack detection, called APLCNet. Frist, Mask R-CNN is applied to extract bottom-level crack information, which is further used in the adaptive feature fusion module (AFFM) to highlight crack attribute and location information. The Mask R-CNN, which is added as a branch in semantic segmentation, extracts crack features and merged these features with high-level features. The hybrid task cascade instance segmentation is also added to semantic segmentation for location information. The AFFM combines the spatial-weighted attention (SAW) mechanism with the channel-weighted attention (CWA) mechanism. The SAW sets the weights of fused features according to their characteristics and CAW creates optimized features by using the weights of SAW features. The CFD dataset and GDPH dataset is first augmented and then used to train the network. The weighted binary cross-entropy loss function was used during the training process. The APLCNet has achieved a 93.53% F1-Score on the CFD dataset.

A computer vision-based inspection method is proposed by (B. Kim & Cho, 2020), which automatically detects multiple damages. The proposed method creates a segmentation mask of damage and cracks using Mask R-CNN with RestNet-101 as a

backbone network to extract major features and Feature Pyramid Network to restore spatial information to detect objects. Region Proposal Network adopts this feature information to indicate damage or crack and detect its location. Another FCN is used to generate a bounding box around the damage and crack in the segmented image. Furthermore, RoIAlign is designed to classify the detected damage. In the final step, the Mask branch estimates the shape of detected damage inside the bounding box. The Mask R-CNN has used the log loss function, L1 loss function, and cross-entropy loss function for classification between damage and background, estimation of the bounding box, and classifying the damage in the bounding box. The Mask R-CNN is pre-trained on the COCO dataset with ResNet-101 and fully trained on 765 concrete images. It achieved 90.41% precision and 90.81% recall for localization and 87.24% precision, and 87.58% recall for segmentation.

2.5.2.6 Segmentation using other Networks

An end-to-end encoder-decoder-based, DL architecture is proposed for pixel-level pavement crack detection (Wenjun Liu et al., 2019). The network consists of Multi-Dilation (MD) module and a Squeeze and Excitation up-sampling module called FPCNet. The FPCNet consists of two 3 x 3 convolutional layers, a ReLU, and a max pooling layer. In addition, binary cross-entropy loss is integrated with a Dice coefficient as a collective loss function, presented in Equation (2.30). The MD module extracts the crack features using four convolutional groups, each group has two 3 x 3 convolutional layers. The crack details obtained by the MD layer are blurry which is further refined through the SE up-sampling module which uses transposed convolution to restore the MD features.

$$L(Y^*, Y) = \frac{1}{N}\sum_{P \in N}(Y_P^* . \lg Y_P + (1 - Y_P^*). \lg (1 - Y_P)) + 1 - \frac{2 \text{ x } TP}{2 \text{ x } TP + FP + FN} \quad (2.30)$$

where Y^* is ground truth pixel, and Y is the predicted pixel. For training and testing purposes, FPCNet is implemented on two different crack datasets CFD and G45. The result achieved on CFD is 97.48% precision, 96.39% recall, and 96.93% F1-Score, while the result achieved in G45 is 95.01% precision, 93.94% recall, and 94.47% F1-Score.

An SDDNet architecture for real-time crack segmentation is developed (Choi & Cha, 2020). The method is proposed to negate background and crack-like features by segmenting the crack in real-time. The network model consists of standard convolutions, a DenSep module (consists of pointwise and depth-wise convolutions), an ASPP module (to gather the surrounding information for better classification), and a decoder module (to produce a fine-grained segmented image). The dataset consists of 200 images (Crack200) collected from different sources and resized to 513 x 513. The dataset is parted into train and test images, 160 and 40 accordingly, respectively. Ground truth images are manually generated by labeling 1 as crack pixel and 0 otherwise. The mIoU is calculated, and a cyclical learning rate policy is adopted for training iterations using Equation (2.31).

$$lr_2 + (lr_1 - lr_2) \times \max(0,1 - X) \times \gamma^{iter} \tag{2.31}$$

where lr_{i} is hyperparameter, $iter$ is training iteration, and γ is a constant value (0.999). The model was pre-trained on the Cityscape dataset and trained from scratch on the Crack200 dataset. The architecture is compared with the DeepCrack model and evaluated on the DeepCrack and Crack200 datasets. The achieved results are Precision 21.3-80.5, Recall 56.1-83.4, F1-Score 30.9-81.9, and mIoU 58.7-84.6. The proposed model has 0.160 million parameters and is eight times faster (processing time) than the DeepCrack model, which has 14 million parameters. It also requires less than 3MB of hardware storage.

Pixel-level multiple damage detection using FCN is proposed (Shengyuan Li et al., 2019). The proposed FCN architecture is fine-tuned with DenseNet-121 for feature extraction. The widely used DenseNet for large-scale images contains a dense block that combines all the features and passes to the next layer. The concatenation alleviates the vanishing gradient problem, strengthens feature propagation, encourages feature reuse, and substantially reduces parameters. A total of 1,375 images are captured with thin cracks and tiny holes. The augmentation method is used to increase the dataset size, and 2,750 images are divided into 80% for the training (includes validation) and 20% for testing. The proposed FCN architecture achieved 98.61% mean pixel accuracy, 84.53% mIoU, and 98.61% PA.

Crack images contain multifaceted information regarding spatial feature extraction, such as complex background, poor texture, and linearity of interference, which causes difficulties in the learning process for crack detection. Therefore, an end-to-end crack segmentation network is proposed, called CrackSeg (Song et al., 2020) to detect road cracks at a pixel level. The network is based on ResNet architecture that consists of multiscale dilated convolution and an up-sampling module. To build the dataset, 8,188 crack images are captured from 14 cities in Liaoning province, China. The dataset is divided into 4,736 training images, 1,036 validation images, and 2,416 testing images, along with AigleRN and CFD datasets. In addition, these images are divided into smaller blocks of size 256 x 256. As a result, the network performed more efficiently than other networks by scoring 98.00% precision, 97.85% recall, 97.92% F-measure and 98.79% mean accuracy.

A ring robot system based on multiple vision cameras controlled by a computer is developed to evaluate bridge pier cracks (K. Jang et al., 2020). The ring robot captures raw images for feature extraction, and control-based image stitching enhanced the image

contrast for better feature extraction, matches extracted features, and removes mismatched features. The semantic segmentation contains modified SegNet with 10 convolutional layers, four max pooling, and four up-sampling layers. A total of 1021 training images are used to train the network. First, the Euclidean distance transformation technique quantifies the detected cracks. Then, the images are converted into grayscale from RGB through the median filter. Next, the resultant is subtracted from the grayscale images to remove the background noise. Afterward, Otsu's method is applied to convert the resultant image into a binary image. Finally, the Skeletonization process is used to extract shape for computing crack length and width. The proposed technique successfully evaluates crack with 90.92% precision and 97.47% recall.

Pixel-level crack detection on dam surface (CDDS) method is proposed (Feng et al., 2020) to inspect hydropower stations safely. The detection process is carried out using an unmanned aerial vehicle that captured 1,000 raw images from the dam surface and for further process, 504 crack images were selected. The detection architecture consists of an encoder-decoder, connected with a skip connection. The CDDS is based on SegNet architecture to extract sparse and dense features. The network used 404 images for training, 50 for validation, and 50 for testing. As a result, the CDDS achieved 80.31% precision, 80.45 % recall, and 79.16% F1-Score.

In (Shengyuan Li & Zhao, 2020), a convolutional encoder-decoder network (CedNet) is designed to segment the crack pixels, eliminate the detected crack distortion, and use a post-processing technique to measure the crack width and orientation. The proposed encoder-decoder network is based on DenseNet-121 which contains an Eltwise layer in a dense block along with a convolutional layer, a pooling layer, a normalization, and a ReLU activation in the encoder part. The decoder part contains an up-sampling layer, convolutional layers, interpolation, and SoftMax activation. The dataset has 372 images

of size 4032 x 3024 captured from the iPhone 7. These images are cropped into 1800 smaller images of size 761 x 569. To train the network, 1500 images are selected from the dataset, randomly while the remaining 300 images are used for testing. The CedNet achieved 98.90% accuracy, 93.58% precision, 94.73% recall, and 93.18% F1-Score. Once the crack pixels are detected, the relations of ratio and field of view are used along with image resolution for measurement. The widths and orientations of cracks are measured using the Euclidean distance transformation and least square fitting algorithm.

A rotation invariant fully convolutional network, called ARF-Crack is proposed (F. C. Chen & Jahanshahi, 2020). The ARF-Crack adopted the DeepCrack network for crack segmentation and active rotating filters are used to encode rotation invariant properties into the network. The ARFs extract the feature from four different orientations (0, 90, 180, 270) in each layer and each rotation has four filters. The generated output fused with a convolutional output of the previous layer at a certain orientation. The network used a summation of regularization loss and prediction loss (cross-entropy loss). The proposed network is trained and evaluated on four different datasets i.e., DeepCrack dataset (Y. Liu et al., 2019), CFD (Shi et al., 2016b), CRACK 500 (Lei Zhang et al., 2016b), and GAPS384 (Eisenbach et al., 2017). The ARF-Crack has achieved a 91.8% average precision score on the DeepCrack dataset.

2.5.2.7 Interpretation of Crack Detection Through Segmentation

Crack image segmentation has outperformed the classification and bounding box methods which only classify or create a rectangular box around the crack in the given image. Crack segmentation produces a label for each crack and non-crack pixel. Several CNN-based network architecture has been proposed for crack segmentation in the last few years. Among these architectures, U-Net, VGG network, and their variants are most

widely adopted for crack segmentation. Although the performance of crack segmentation networks is outclassed, still several challenges are open to researchers.

Table 2.2 illustrates the summary of the main findings of the related work as discussed in this section.

Table 2.2: Related work summary for crack segmentation

References	Architecture	Dataset	Loss function
(Dorafshan et al., 2018b)	CNN (AlexNet)	SDNET2018 Dataset	CE
(Fan et al., 2018a)	DCNN	AigleRN Dataset CrackForest Dataset	CE - Weighted decay
(Vu & Duc, 2019)	DCNN	Self-Collected Images	CE
(Y. Ren et al., 2020)	DCNN	Self-Collected Images	Focal loss
(David Jenkins et al., 2018)	U-Net	CrackForest Dataset	CE
(Ji et al., 2018)	U-Net	Self-Collected Images	CE
(Cheng et al., 2018)	U-Net	AigleRN Dataset CrackForest Dataset	CE - Weighted decay
(Z. Liu et al., 2019)	U-Net	Self-Collected Images	Focal loss
(Konig et al., 2019)	CNN (VGG-16)	CrackForest Dataset	CE and Dice Loss
(Lau et al., 2020)	U-Net (ResNet-32)	CrackForest Dataset Crack500 dataset	Dice
(Lingxin Zhang et al., 2020)	U-Net	Self-Collected Images Collected from the Internet	Dice
(Huyan et al., 2020)	U-Net	Self-Collected Images	CE
(Dong et al., 2020)	U-Net-ResNet with PAM	DeepCrack Cls-dataset	WCE
(J. Liu et al., 2020)	YOLOv3+ U-Net (ResNet-32)	Self-Collected Images CrackForest Dataset	Dice
(X. Yang et al., 2018)	CNN (VGG-19)	Self-Collected Images Collected from the Internet	CE
(Fei et al., 2019)	CNN (VGG16)	3D pavement Images CrackForest Dataset	BCE
(A. Zhang et al., 2017)	CNN (CrackNet)	3D pavement Images	CE

References	Architecture	Dataset	Loss function
(Y. Liu et al., 2019)	FCN + VGG16	Self-Collected Images	CE
(X. Zhang et al., 2019)	CNN (VGG16)	CrackForest Dataset SDNET2018 Dataset Tomorrows Road Infrastructure Monitoring, Management Dataset	WCE
(M. Wang & Cheng, 2019)	CNN (VGG-19)	Self-Collected Images	CE
(J. S. Lee et al., 2020)	CNN (VGG16)	Collected from the Internet	WCE
(T. Chen et al., 2020)	CNN (VGG16)	Self-Collected Images	CE
(G. Li et al., 2020)	VGG + Inception	Self-Collected Images	CE
(L. Xu et al., 2020)	VGG + CNN (LNet and DNet)	KolektorSDD Dataset	BCE
(A. Zhang et al., 2018)	CNN (CrackNet-II)	3D pavement Images	CE
(Kaige Zhang et al., 2018)	CNN (AlexNet)	Self-Collected Images	CE
(Ni et al., 2019)	GoogleNet + CDN	Self-Collected Images	BCE
(Bang et al., 2019)	ResNet-152	Self-Collected Images	CE
(D. Lee et al., 2019)	CSN + CNN	Microsoft COCO Dataset	CE
(Yamane & Chun, 2020)	Mask R-CNN	Self-Collected Images	CE
(Kang et al., 2020)	Faster R-CNN + ResNet-50	Self-Collected Images	CE
(Y. Zhang et al., 2020)	ALPCNet (Mask R-CNN and AFFM)	GDPH Dataset	WCE
(B. Kim & Cho, 2020)	Mask R-CNN	Self-Collected Images	Log Loss, Regularization Loss, and CE Loss
(Wenjun Liu et al., 2019)	FPCNet	CrackForest Dataset G45 Crack Dataset	CE Dice
(Choi & Cha, 2020)	SDDNet	Cityscape dataset Crack200 dataset	mIoU
(Shengyuan Li et al., 2019)	FCN + DenseNet-121	Self-Collected Images	CE
(Song et al., 2020)	CrackSeg (ResNet)	Self-Collected Images	CE
(K. Jang et al., 2020)	SegNet	Self-Collected Images	CE

References	Architecture	Dataset	Loss function
(Feng et al., 2020)	SegNet	Self-Collected Images	Dice
(Shengyuan Li & Zhao, 2020)	CedNet (DenseNet-121)	Self-Collected Images	CE
(F. C. Chen & Jahanshahi, 2020)	ARF-Crack	GAPS384	Regularization Loss and CE Loss

2.6 Summary

The use of DL innovations is driving exciting breakthroughs in the field of medical, engineering, finance, mathematics, information technology, and many more. It has been introduced to many applications which are becoming part of our daily life such as disease diagnoses, stock market, self-driving cars, and information retrieval. One of the applications of DL in civil engineering is crack detection from images. Cracks are detected in these images by classification, segmentation, or bounding box method. Numerous DL-based crack detection methods are discussed in the literature section that resolves the limitations of IPT and other ML techniques.

Crack detection using image classification only detects the presence of a crack in the image. It does not provide specific crack information. Furthermore, the bounding box technique was also adopted by a few researchers but the issue of obtaining the crack information remained unsolved because the bounding box only creates a rectangular box around the crack in the image. Moreover, to identify the crack pixels in the image, several semantic segmentation techniques are proposed which generate the segmented mask of crack and non-crack pixels. To attain crack segmentation accuracy, different CNN networks are implemented. These networks become biased due to class imbalance between crack and non-crack pixels. Therefore, the networks tend to segment non-crack pixels more than crack pixels. This limits the network performance, and the network fails to achieve high accuracy for both classes.

A proper balancing method is required with a crack segmentation network that generates segmented masks with high accuracy for both classes. The method must incorporate the class imbalance issue, and accurately detection of boundary pixels and pixels belonging to spot, stain, and dark intensities, as they are hard to predict under normal circumstances.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

In chapter two, a detailed literature review of crack detection methods is presented and discussed. The DL-based techniques utilize the advantage of CNN architectures to classify the crack image or generate the segmentation masks based on crack and non-crack pixels. They are the most efficient and accurate in terms of determining the cracks over other ML techniques and image processing methods. They have prominent results as compared to other ML techniques. However, these methods have some common issues and limitations for crack segmentation applications. These recent studies have adopted patch-level learning and produced an aggregate result. The variation in the number of crack and non-crack pixels introduced the class imbalance problem, which acquaints biasness in the training process. Furthermore, dark intensity regions other than the crack region, black spots, and stains in the training images are the cause of less segmentation accuracy of crack regions. Most of the methods discussed in Section 2 have simply used the cross-entropy loss function or its variants which evaluates the class predictions for each pixel individually. Similarly, the weighted cross-entropy loss function provides extra help to the minority class, but the global weight is not an optimal value to be used. The dice coefficient causes instability and only considers the foreground, which is non-crack pixels. The distance transformation (DT) map assigns weights to both over and under-segmented pixels causing an increase in false occurrence in both cases. To address all the above-mentioned problems, a pixel-level crack segmentation method is proposed along with a local weighting factor and difference transform map into the loss function. The proposed method increases the crack pixels segmentation accuracy, whereas the local weighting factor balances the difference between crack pixels and non-crack pixels. Meanwhile, the difference transform map helps to upgrade or degrade the corresponding pixels accordingly and increases the chance of occurrence of both classes as a true pixel.

It is an automated end-to-end DL process where the input of the network is in the form of grayscale images and the output is a predicted mask. In this chapter, the employed methodology is discussed concerning the objectives of this thesis.

3.2 Proposed Crack Segmentation Framework

The proposed crack segmentation framework is shown in Figure 3.1. The framework is divided into four different processes. The first part consists of the collection of images, pre-processing, and partition of these crack image datasets into three subsets for training, validation, and testing. The second part consists of the training process, i.e., the training images and validation images are used to train the convolutional neural networks along with various loss functions. This trained network is evaluated using test images by evaluation metrics in the third part. The fourth and last part uses the generated mask of test images to measure the crack characteristics.

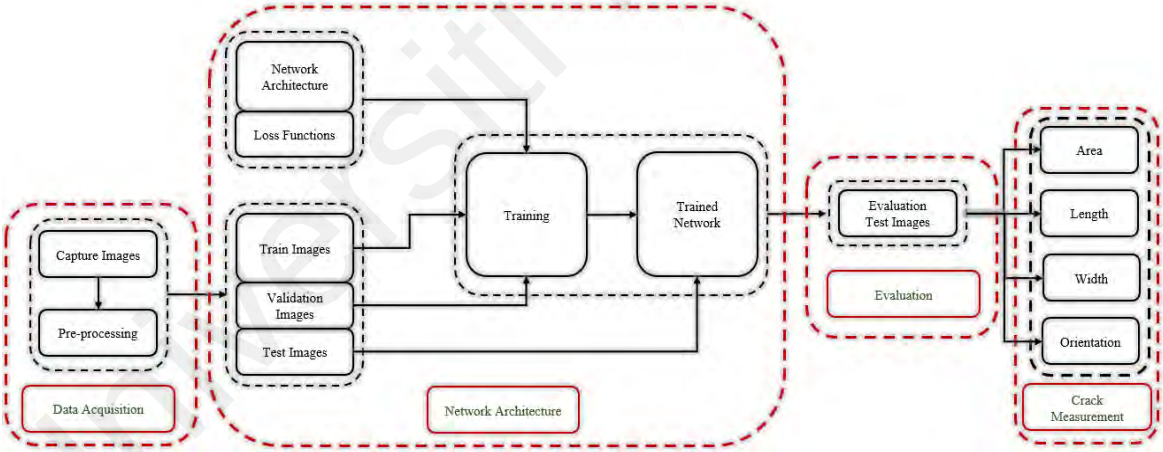

Figure 3.1: Overview of the proposed crack pixel segmentation framework

3.3 Dataset

Dataset is one of the major factors determining the performance of CNN. The performance of CNN models varies with the number of images present in the dataset and the environment under which these images are captured. Many images are required for the network to perform satisfactorily, therefore, augmentation techniques are used to enhance the images in the dataset. A small dataset results in overfitting for the deeper

networks, whereas a large dataset avoids the overfitting issue (Hesamian et al., 2019). Dataset needs to be balanced (an equal number of crack and non-crack images/pixels in an image) because an imbalanced dataset can cause degradation in the accuracy of the network. It is important to note that due to the lack of publicly available crack datasets, one could not quantitatively perform fair comparisons between existing methods. The crack image dataset can steer the network performance because there is no standard crack image dataset that can be used for architecture comparison. The publicly available datasets have variations due to several factors such as type of structure, condition of the structure, time of capturing an image, a device used to capture the images, and many more. It has been observed from Section 2, that most of the authors used a self-collected dataset for both classification and segmentation and the most used publicly available dataset is CFD. The frequent use of self-collected images is due to the lack of a standard dataset for crack detection. There are several datasets but not publicly available and were created by the authors themselves.

Therefore, to train and evaluate the performances of networks and loss functions, a crack images database is created by capturing a total of 453 crack images at the University of Malaya, Malaysia. The dataset which has diverse orientations in terms of crack width and position is called the Multi Structure Crack Image (MSCI) dataset. The capturing task is conducted by using Canon EOS 1300D camera model. For better results and focused images, camera resolution was set at 5184 x 3456. Only those images of cracks are captured which lay at a distance of 1 to 3 feet between the source (crack) and the camera. Meanwhile, the vertical distance was 7 to 8 feet high from the ground. This crack dataset is a typical example of an unbalanced class problem; the images contain a disproportionate number of crack and non-crack pixels.

3.3.1 Dataset Preparation

To use the captured images with the proposed framework, the original images are divided into multiple RGB images of 512 x 512 size. These images are sorted and only those images which contain cracks are selected for further processing, the images which do not contain cracks are eliminated and not considered for the experiment. The selected images are further processed by eliminating every second pixel to lower the resolution to 256 pixels per row/column as shown in Figure 3.2. Reducing image size helps fast processing and meets the capability level of available hardware, i.e., GPU.

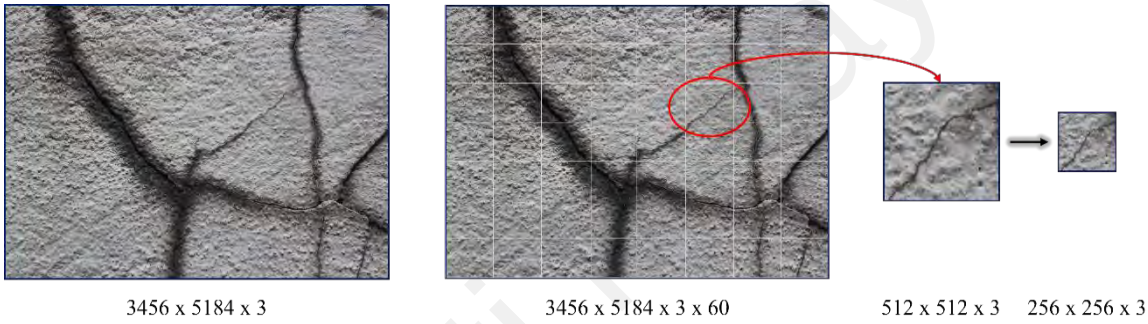

Figure 3.2: Image resizes

Therefore, a total of 2500 crack images are selected for the experiment. The selected images are manually labeled and cross-checked by experts to generate ground truth masks using the MATLAB image labeler application as shown in Figure 3.3. The labelled mask is further cross-checked and verified by experts from civil engineering and image processing fields.

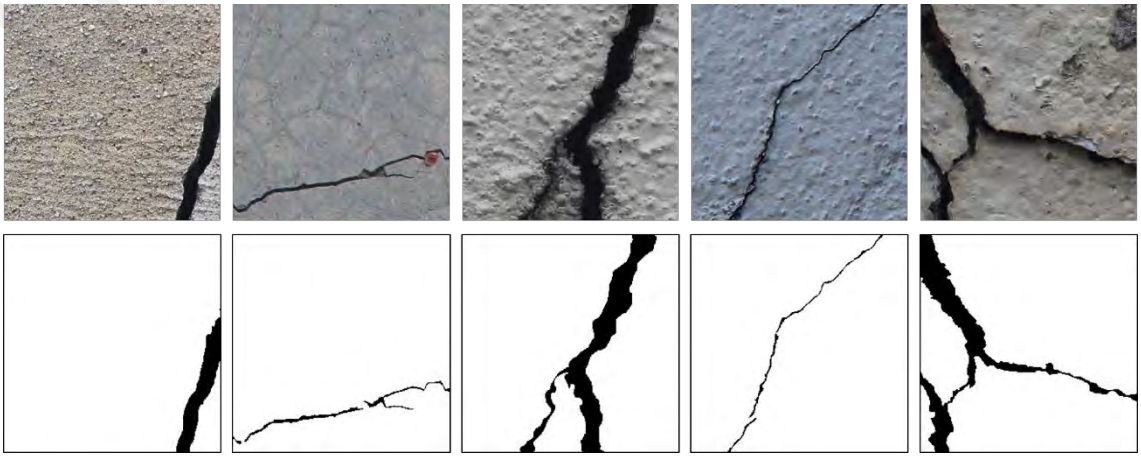

Figure 3.3: Examples of color images (first row) and labels (second row)

3.3.2 Image Pre-processing

The feature extraction process of DL algorithms depends on the refinement of data. The neural network learns from the image dataset. To build an efficient trained network, appropriate and effective image processing techniques need to be applied to the image dataset. The image contains useful information which can be exposed through IPTs. The IPTs can produce output in the form of another image, attributes, or sympathetic results. The output mainly depends on the level of processing applied to the image. Crack images contain several other objects such as stains, spots, and dark intensity regions in terms of noise. These noisy objects can be removed either through image processing-based denoising techniques (Makinen et al., 2020) or DL denoising techniques (Kai Zhang et al., 2018). Therefore, in this research, two strategic image processing techniques are adopted to enhance the characteristic of both crack and non-crack pixels for effective learning.

The first image processing operation is to adopt the single-channel image network by converting the three-channel RGB images into single-channel grayscale images. The RGB color image is split into three color channels, i.e., red, green, and blue channels. The contrast difference between crack and non-crack pixels for single-channel grayscale images is better than in RGB images (Fan et al., 2018b; K. Jang et al., 2020). Therefore, the original RGB images are converted into a grayscale image by using Equation (3.1) (Sapkal et al., 2008).

$$I_{gray} = (0.299 \times R) + (0.587 \times G) + (0.114 \times B) \tag{3.1}$$

where R, G, B are the red, green, and blue channels with the corresponding contribution in percentage, respectively. The output grayscale images are shown in Figure 3.4 (second row).

The second image processing technique applied to the image dataset is to increase the contrast between the crack and non-crack regions. The grayscale images are used, and their contrast is increased by mapping the values of the input intensity image to new values such that, 1% of the data is saturated at low and high intensities of the input data. The output of contrast enhancement images is shown in Figure 3.4 (third row). The implementation of the grayscale and contrast enhancement function is performed using MATLAB (MathWorks, 2020).

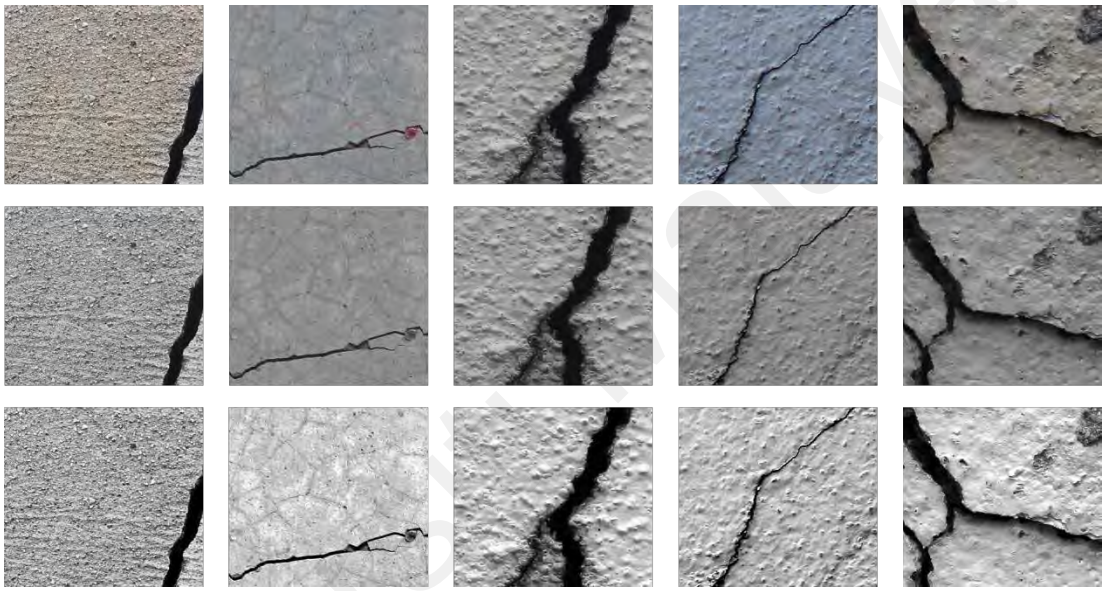

Figure 3.4: Examples of RGB images (first row), grayscale images (second row) and contrast enhancement images (third row)

3.3.3 Labelling

In supervised learning, the network requires two types of inputs for training, original images, and their corresponding labelled mask. Collecting and editing data (crack images) is one of the important aspects of DL because only correct labeled data will help in the learning process, leading to superior performance. Annotation or labeling is one of the hardstand times taking part in crack pixel segmentation. A ground-truth mask along with the original image is needed to feed the network for learning. The mostly used labeling tools are LabelMe (Ji et al., 2018), (M. Wang & Cheng, 2019), (G. Li et al., 2020), Linear image annotation tool (Bang et al., 2019), Photoshop (Shengyuan Li et al., 2019),

(Shengyuan Li & Zhao, 2020) and the Affinity photo tool (Choi & Cha, 2020). Similarly, a heat map is another way of labeling crack pixels to visualize through the different color ranges from warm to cool. The color range is shown through the color spectrum, the crack region as the targeted region, and the other regions of images can be shown with low-intensity colors.

In this research, the MATLAB-Image Labeler toolbox is used to label the crack and non-crack pixels. The toolbox contains several user-friendly drawing tools such as polygon, smart polygon, assisted freehand, brush, erase, and flood fill to create and mark the region of interest of any arbitrary shape (rectangular, polyline, pixel, and polygon). To mark the crack image dataset, two labels are created: namely CrackPixels and Non-CrackPixels. The CrackPixels label contains the information of all those pixels which are marked as crack pixels and the NonCrackPixels label contains the information of all those pixels which are marked as non-crack pixels. These labels determine the location of the original image, coordination of pixels that belong to their respective class, color information that represents each class, and pixel-ID. The crack and non-crack pixels are presented with two different and unique colors for differentiation as shown in Figure 3.5. The crack and non-crack pixels are labeled using assisted freehand and flood fill tools.

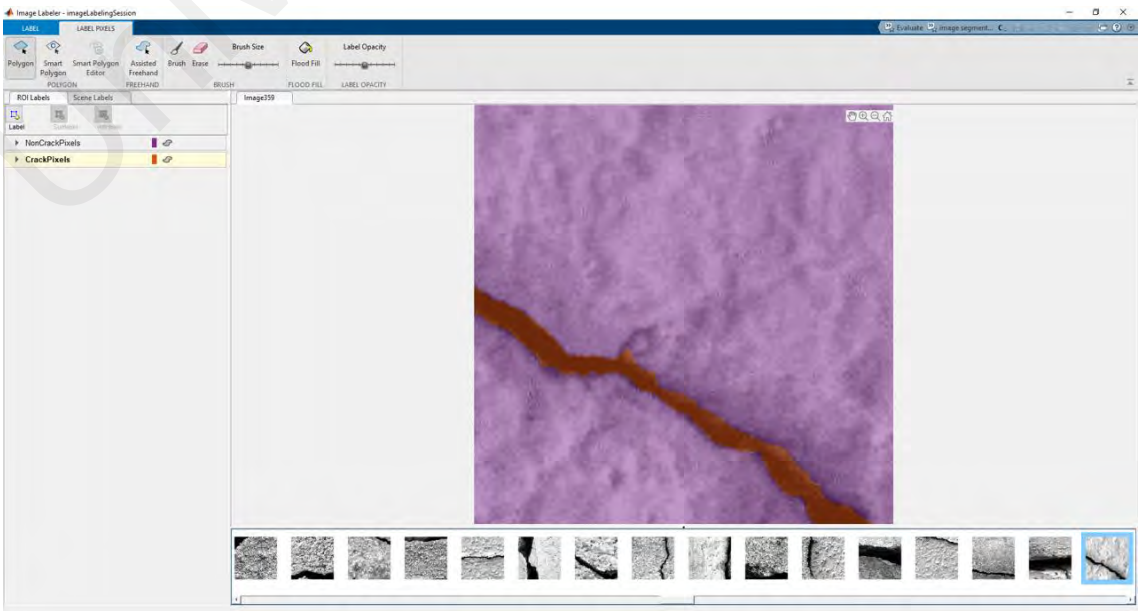

Figure 3.5: Image labeler toolbox

The transition region between crack and non-crack is the hardest area to label as it is very confusing and required zooming to be marked properly, while the other regions are easy to mark. These regions are time-consuming as well. After labelling, the ground truth mask of each image is generated in the form of a binary image, as shown in Figure 3.6.

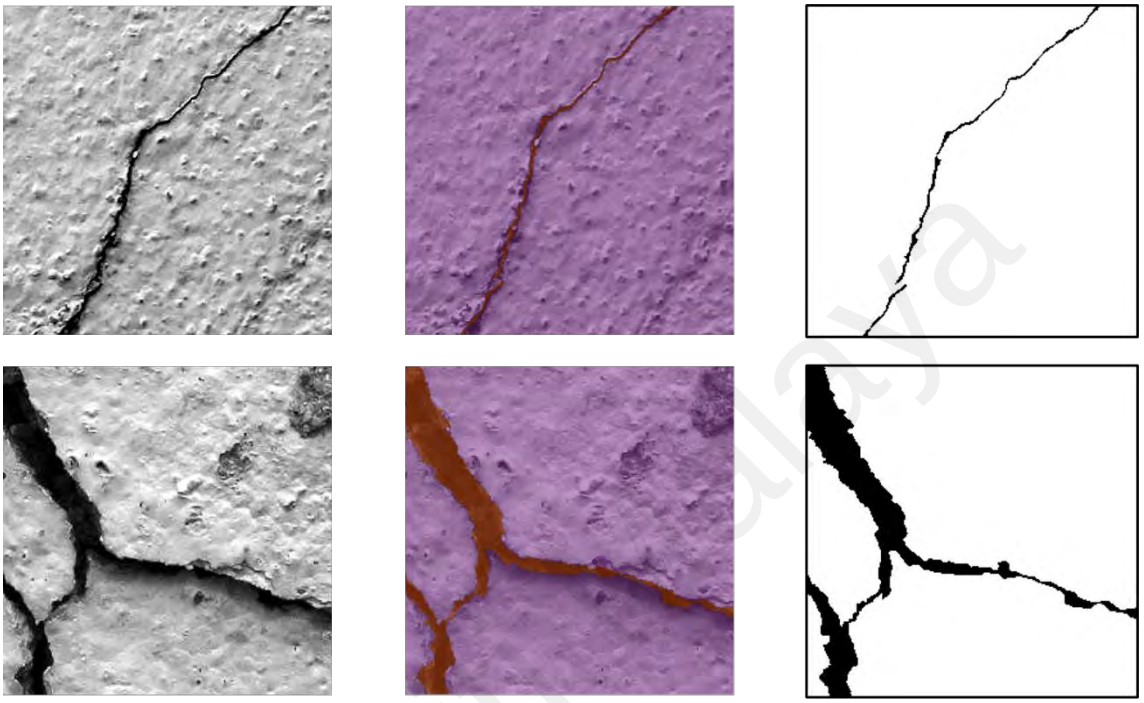

Figure 3.6: Labeled and ground truth mask

3.4 Crack Segmentation Network Architectures

Semantic segmentation classifies crack pixels and non-crack pixels in the image according to their respective class. For crack segmentation applications, several DCNNs with different structures are proposed for crack pixel classification tasks. Recent research on crack segmentation utilizes encoder-decoder networks which have symmetric structures. The encoder part extracts the contextual features information from the inserted input images or feature maps from previous layers and the decoder part restores the dimensional features information from the latent space. To illustrate the performance of the proposed crack segmentation network architecture and loss function (local weighting factor and difference transform matrix) on imbalanced datasets, several well-known, widely adopted and state-of-the-art network architectures such as FCN (J. Long et al.,

2015), U-Net (Ronneberger et al., 2015), SegNet (Badrinarayanan et al., 2017), and DeepLabv3+ (L.-C. Chen et al., 2018), are trained in an end-to-end, pixel-to-pixel manner for crack image segmentation. The above-mentioned segmentation network architectures are discussed below.

3.4.1 FCN Architecture

An end-to-end, pixels-to-pixels, state-of-the-art semantic segmentation network architecture, is proposed (J. Long et al., 2015). The key insight is to build an efficient inference and learning network that is fully convolutional. The network performed pixel-by-pixel prediction and generates an output segmented mask with the corresponding input size.

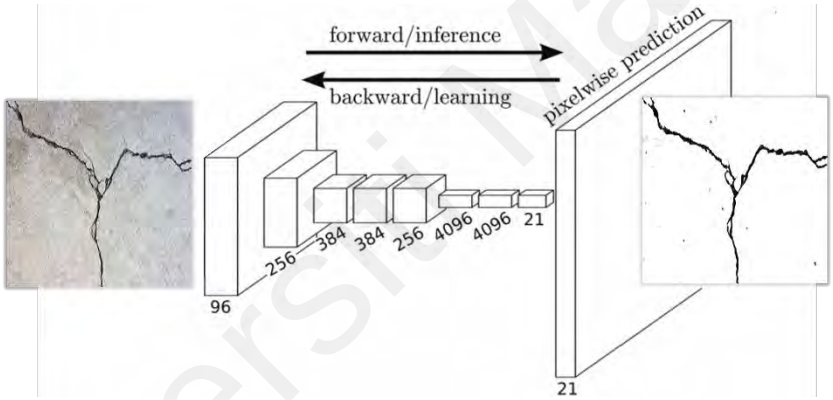

Figure 3.7: FCN architecture (J. Long et al., 2015)

Figure 3.7 shows the FCN network that employs convolutional, pooling, and up-sampling layers by replacing fully connected dense layers. The FCN network also utilizes the advantage of skip connections which improves the performance by using the feature maps from previous layers. The network has three different versions FCN-32s, FCN-16s, and FCN-8s, based on the up-sampling layer that generates the output mask. The FCN-32s up-samples at stride 32, do not use skip connections and use a single step for pixel prediction. The FCN-16s contain finer details than FCN-32s. It has deeper information as it combines the prediction from the pool4 layer and the final layer. The FCN-8s have the

deepest spatial information because the prediction is based on pool3 at stride 8 shown in Figure 3.8 which provides precise boundary predictions.

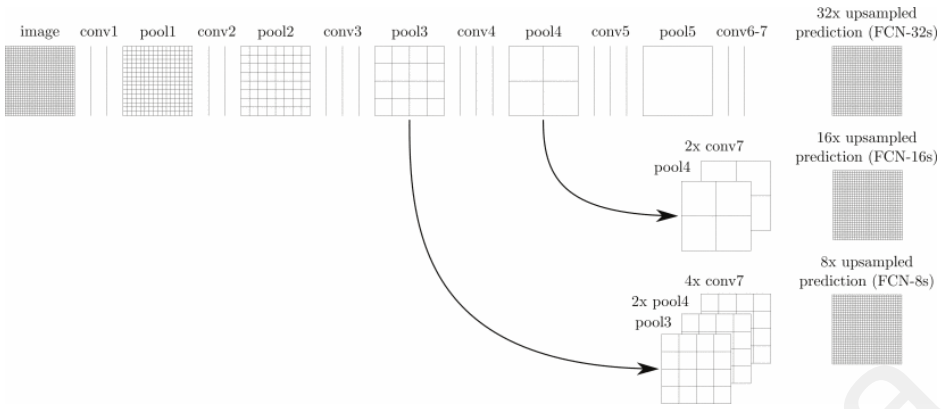

Figure 3.8: Fusing for FCN-16s and FCN-8s (J. Long et al., 2015)

3.4.2 U-Net Architecture

An end-to-end, U-shaped CNN architecture is shown in Figure 3.9, proposed in (Ronneberger et al., 2015). The U-Net architecture is developed for medical image segmentation and due to its similar structure to vain/retinal blood vessels, this architecture also performed well on crack image segmentation. The architecture has 23 convolutional layers distributed among contracting and expanding paths. They are used to extract the context and precise localization information from the feature maps. The contracting path has convolutional layers along with ReLU activation function and max pooling operation whereas, the expanding path consists of up-sampling operation along with convolutional

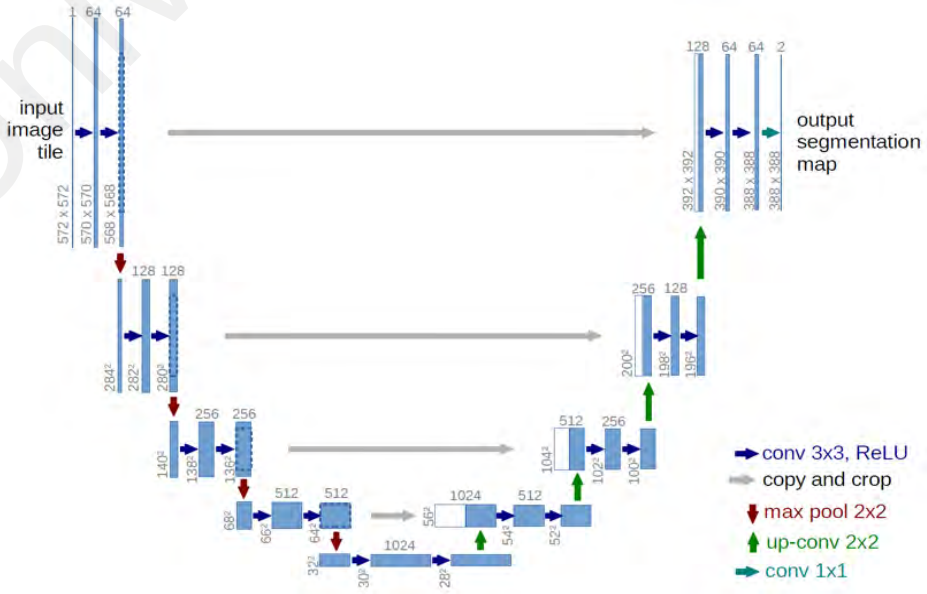

Figure 3.9: U-Net architecture (Ronneberger et al., 2015)

layers and ReLU activation function. The network with the proper augmentation technique is capable to perform well with a small dataset.

3.4.3 SegNet Architectures

A fully CNN was proposed for pixel-wise image segmentation called SegNet (Badrinarayanan et al., 2017). It is an encoder-decoder-based CNN with pixel-wise classification as an output layer shown in Figure 3.10. The encoder network consists of 13 convolutional layers and each layer corresponds to decoder layers. The convolutional layers in each encoder block applied batch normalization, ReLU activation function, and Max-polling operation. The decoder block uses an up-sampling layer, convolutional layer, batch normalization, and ReLU activation function. The output from the last decoder block is inserted into the SoftMax classifier which helps to determine the probability of each pixel to associate the pixel to their maximum likelihood class. The SegNet architecture only stores max pooling indices therefore it requires less memory. The decoder has 64 feature maps, which slow down the training process but provide more flexibility to achieve higher training accuracy.

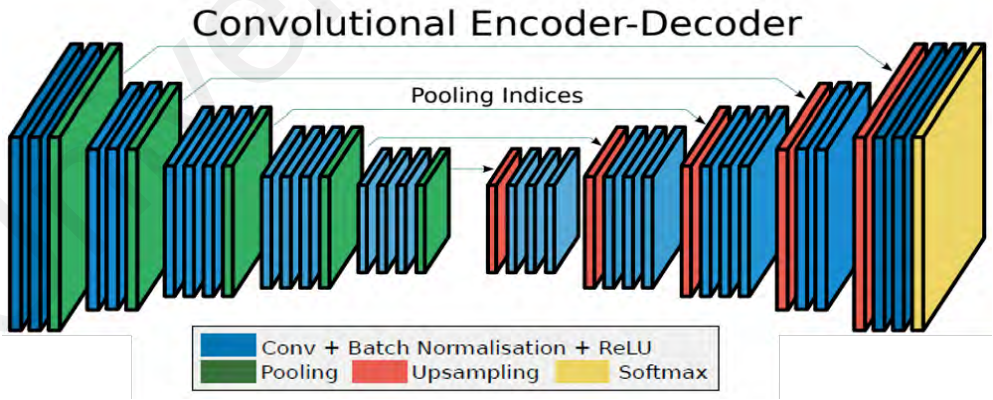

Figure 3.10: SegNet architecture (Badrinarayanan et al., 2017)

3.4.4 DeepLabv3+ Architectures

DeepLabv3+ is an encoder-decoder architecture proposed in (L.-C. Chen et al., 2018), as shown in Figure 3.11. It is an extended version of DeepLabv3 by adopts an effective decoder module to refine the segmentation output. The encoder extracts the essential

information such as the presence of objects and their location, whereas Atrous-convolution controls the feature resolution and adjusts the filter's field-of-view to capture multiscale information. The decoder part uses the extracted information to generate an output with the size of the original input.

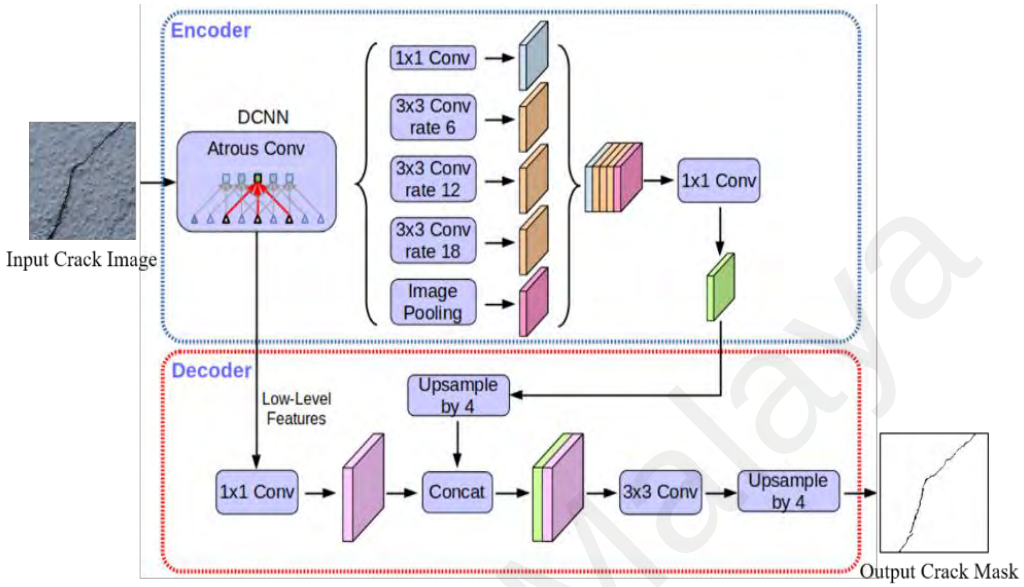

Figure 3.11: DeepLabv3+ architecture (L.-C. Chen et al., 2018)

3.4.5 Proposed Crack Segmentation Network Architecture

A deep fully CNN for crack segmentation is proposed, shown in Figure 3.12. The proposed network consists of an input layer, encoder block, base layer, decoder block, reshape layer, and output layer. The input layer contains the training and validation

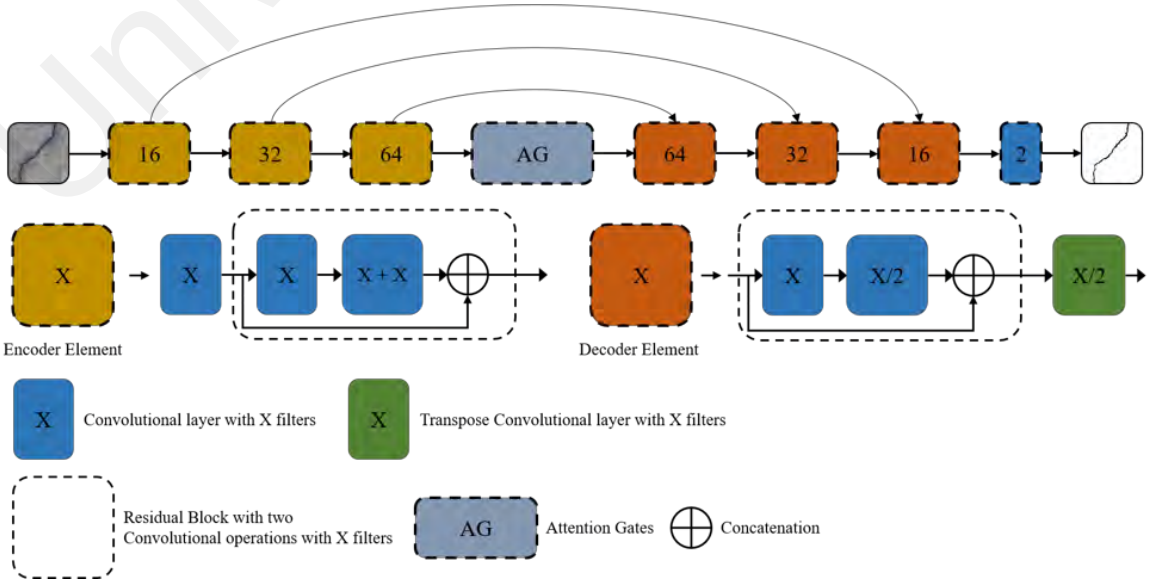

Figure 3.12: Crack segmentation network architecture

images use to feed the network. The network is designed to be trained on grayscale images (256 x 256 x 1) that reduces the network complexity by using the single-channel image. This layer understands the input data and has no learnable parameters. The network has 1,691,322 learnable parameters collectively from all convolution operations in the network. Each input image passes through the encoder block, base layer, decoder block, reshape layer, and SoftMax function in the last layer to generate a segmentation mask. The encoder block consists of three encoder elements, where each encoder element contains a convolutional block and a residual block. The convolutional block is further consisting of convolution operation, activation function, and batch normalization as shown in Figure 3.13. Each encoder element is also connected to either the input layer or previous encoder elements and its respective decoder element through skip connections.

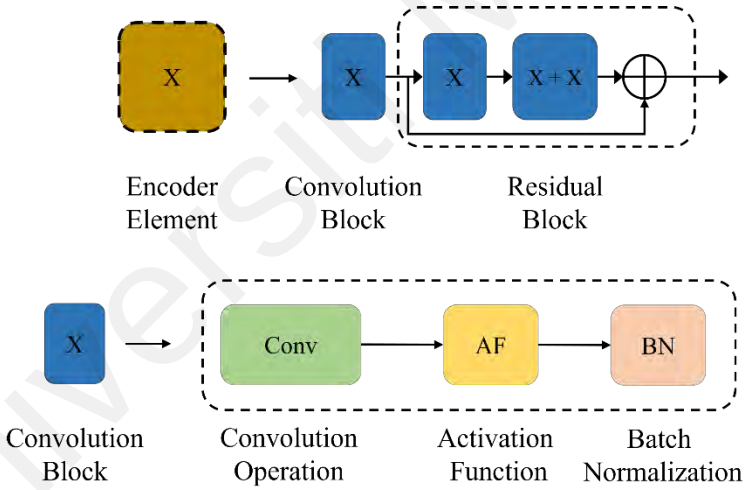

Figure 3.13: Encoder element

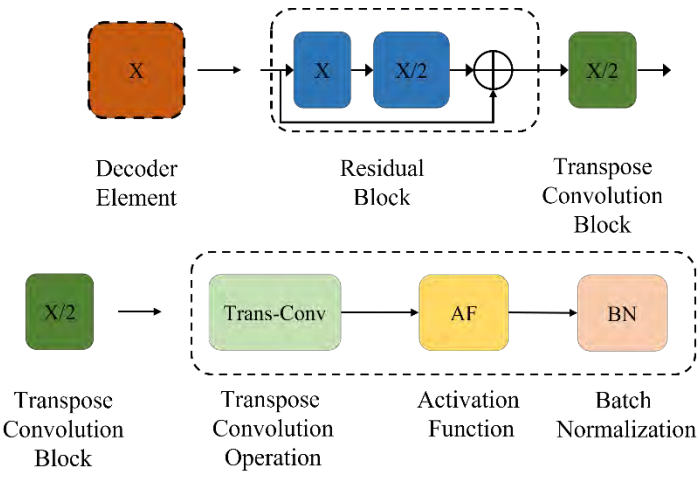

Figure 3.14: Decoder element

A base layer consists of Additive Attention Gate (AAG) module that connects the encoder block with the decoder block by up-sampling the feature maps. The AAG consists of four convolutional blocks, attached parallel to each other. The output of these four convolutional blocks is concatenated and inserted into the SoftMax function followed by a convolution operation to adjust the number of filters for feature maps to be inserted in the decoder block. The decoder block also consists of three decoder elements, where each decoder element contains a residual block followed by a transpose convolutional block. The transpose convolutional block is further consisting of transpose convolution operation, activation function, and batch normalization as shown in Figure 3.14. The reshape convolutional layer consists of a single convolutional operation with two filters of 1x1 filter size. This layer reshapes the feature maps extracted from the last decoder element to 256 x 256 x 2, where 256 is the height and weight of the original image, and 2 represents the number of output classes to be segmented. The proposed network has 2^x (where $x = 3,4,5,6,7,8,9,10$) number of filters in each convolutional operation except reshape convolutional operation. Every convolution operation is followed by batch normalization and activation function. The details of the proposed architecture are tabulated in Table 3.1.

Table 3.1: Crack segmentation network architecture

Layer Name	Filter Size	No. of filter	Stride	Padding	Activation	Operational layers
Input Image	-	-	-	-	256 x 256 x 1	-
Encoder_1_Conv	3 x 3	16	1,1	Same	256 x 256 x 16	ReLU, Batch Normalization
Encoder_1_RB	3 x 3	16	1,1	Same	256 x 256 x 16	ReLU, Batch Normalization
	3 x 3	32	1,1	Same	256 x 256 x 32	ReLU, Batch Normalization
Connecting Layer_1	-	-	-	-	256 x 256 x 48	Depth Concatenation
Encoder_2_Conv	3 x 3	32	2,2	Same	128 x 128 x 32	ReLU, Batch Normalization
Encoder_2_RB	3 x 3	32	1,1	Same	128 x 128 x 32	ReLU, Batch Normalization
	3 x 3	64	1,1	Same	128 x 128 x 64	ReLU, Batch Normalization

Connecting Layer_2	-	-	-	-	128 x 128 x 96	Depth Concatenation
Encoder_3_Conv	3 x 3	64	2,2	Same	64 x 64 x 64	ReLU, Batch Normalization
Encoder_3_RB	3 x 3	64	1,1	Same	64 x 64 x 64	ReLU, Batch Normalization
	3 x 3	128	1,1	Same	64 x 64 x 128	ReLU, Batch Normalization
Connecting Layer_3	-	-	-	-	64 x 64 x 192	Depth Concatenation
Attention_Gate	3 x 3	2	1,1	Same	64 x 64 x 2	ReLU, Batch Normalization
	3 x 3	2	1,1	Same	64 x 64 x 2	ReLU, Batch Normalization
	3 x 3	2	1,1	Same	64 x 64 x 2	ReLU, Batch Normalization
	3 x 3	2	1,1	Same	64 x 64 x 2	ReLU, Batch Normalization
	-	-	-	-	64 x 64 x 8	Depth Concatenation
	-	-	-	-	64 x 64 x 8	SoftMax
	3 x 3	192	1,1	Same	64 x 64 x 192	ReLU, Batch Normalization
	-	-	-	-	64 x 64 x 192	Addition
Connecting Layer_4	-	-	-	-	64 x 64 x 384	Depth Concatenation
Decoder_1_RB	3 x 3	128	1,1	Same	64 x 64 x 128	ReLU, Batch Normalization
	3 x 3	64	1,1	Same	64 x 64 x 64	ReLU, Batch Normalization
Connecting Layer_5	-	-	-	-	64 x 64 x 256	Depth Concatenation
Decoder_1_Trans_Conv	3 x 3	64	2,2	Same	128 x 128 x 64	ReLU, Batch Normalization
Connecting Layer_6	-	-	-	-	64 x 64 x 160	Depth Concatenation
Decoder_2_RB	3 x 3	64	1,1	Same	128 x 128 x 64	ReLU, Batch Normalization
	3 x 3	32	1,1	Same	128 x 128 x 64	ReLU, Batch Normalization
Connecting Layer_7	-	-	-	-	128 x 128 x 192	Depth Concatenation
Decoder_2_Trans_Conv	3 x 3	32	2,2	Same	256 x 256 x 32	ReLU, Batch Normalization
Connecting Layer_8	-	-	-	-	256 x 256 x 80	Depth Concatenation
Decoder_3_RB	3 x 3	32	1,1	Same	256 x 256 x 32	ReLU, Batch Normalization
	3 x 3	16	1,1	Same	256 x 256 x 16	ReLU, Batch Normalization
Connecting Layer_9	-	-	-	-	256 x 256 x 96	Depth Concatenation
Decoder_3_Trans_Conv	3 x 3	16	2,2	Same	256 x 256 x 16	ReLU, Batch Normalization
Convo_Reshape	1 x 1	2	1,1	Same	256 x 256 x 2	SoftMax
Pixel classification layer	-	-	-	-	256 x 256 x 2	Loss Function

3.4.5.1 Convolutional Layers

The convolutional layer is the main block of CNN architecture which convolves the input image or feature map from the previous layer with filters to generate new feature maps. This layer has learnable parameters such as weights and biases. This layer also contains different filters to extract information from the feature maps or input images. These filters detect edges, shapes, and patterns from the input image. A dot product between each input image and filter is performed, followed by summing each dot product output, and finally, a bias is added. Bias can be configured according to network requirements. The convolutional layer reduces the computational cost by reducing the input size.

$$Z_i^l = \sum_{J=1}^{K^{l-1}} W_{i,j}^l * Z_j^{l-1} + B^l \quad (3.2)$$

Equation (3.2) shows the operation of the convolutional layer where Z_i^l and Z_i^{l-1} are the outputs of the current layer and previous layer respectively. $W_{i,j}^l$ and B^l represents weights and biases.

Convolutional layers are a set of N learnable convolutional filters which accept M input crack training square images, $I_{i=1}^M$. The first layer has one convolutional layer with 16 filters of 3 x 3 size, F_{3x3}^{16}. The second and third layer has the same layer configuration with F_{3x3}^{32} and F_{3x3}^{64} filters respectively. The "padding" is set to "same" and "stride" of "1 x 1". This setting preserves the image information for generating the output image of the required size. The size of the filters does not affect the training process, but the number of filters influences the learning process. Therefore, we use a different number of convolutional filters in every layer to extract the unique features in every convolutional layer. The last convolutional layer has 2 filters F_{1x1}^2 to reduce the features map size and get the output mask of the same size as of input image. The only class of interest are crack

pixels and non-crack pixels; therefore, the output of the last convolutional layer generates a segmented mask of 256 x 256 for 2 classes $O^{2}_{256x256}$.

3.4.5.2 Residual Block

The concept of the residual block for image recognition is introduced in (He et al., 2016a). An increase in the number of deep neural network layers will start decreasing the accuracy of the network and this phenomenon is called degradation. This degradation may conclude that the shallower networks have better learning and network performance than deeper networks. Therefore, to solve this degradation problem, a skip connection is introduced which learns the difference between input and true output and the layer is learning residual. The residual block shown in Figure 3.15, solved the degradation problem of deep networks.

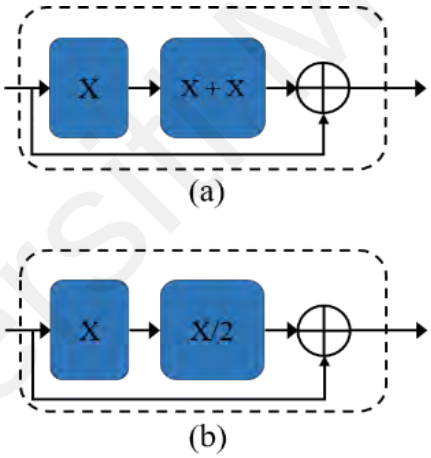

Figure 3.15: Residual blocks

In our proposed crack segmentation network, we use residual blocks with every convolutional and transpose convolutional layer to extract features at two different levels. Each residual block has two convolutional operations with a different number of filters which significantly extract two different levels of feature in the residual block and also speed up the training process. The number of filters in the residual convolutional operation is linked with precursor or successor convolutional operations. In the encoder residual block, the first convolutional operation has X filters and 2X filters in the second convolutional operation shown in Figure 3.15(a). Similarly, the residual block on the

decoder side, has X filters in the first convolutional operation and X/2 filters in the second convolutional operation as shown in Figure 3.15(b). The convolutional stack in the residual block optimizes the original unreferenced features and identifies the optimal features by a stack of nonlinear layers. The shortcut connection in the residual block performs the identity mapping and their outputs are concatenated to the outputs of stacked layers i.e., $F(x) \oplus x$.

3.4.5.3 Additive Attention Gate Module

A soft-attention module called AAG is shown in Figure 3.16, is implemented as a base block in the proposed crack segmentation network which has important features, i.e. (i) to enhance the network ability that automatically learns to extract multi-level refine crack details, (ii) to focus on relevant crack regions to eliminate the use of localization module and (iii) to connect the encoder with the decoder. The last encoder element inputs the feature map to the AAG module, which consists of four parallel convolution layers with filter sizes 3, 5, 7, and 9. The output feature from these convolutional layers is concatenated and passed through a SoftMax activation which exponentially normalized the attentive feature maps and produces relevant coefficient features. The output from the activation function is inserted into a 1 x 1 cascaded convolutional layer to set the dimensions of the attentive map. The resultant is added elementwise in the input feature map as a spirit of residual connections to reduce the learning complexity of the attentive map.

$$x_b' = \left(\left(x_b; \theta^4_{3,5,7,9}\right) \oplus \sigma; \theta^1_1\right) + x_b \tag{3.3}$$

where x_b' is the attentive feature map, x_b is the input feature map at base block b, σ represents the SoftMax function used to normalize the attentive score, ; represents convolution operation, and $\oplus$ represents the concatenation operation. The $\theta^4_{3,5,7,9}$ represents the four parallel convolutional layers (the value of superscripts determines the

number of convolutional operations) with different filters size (the value of subscripts determines the filter size of each convolutional operation).

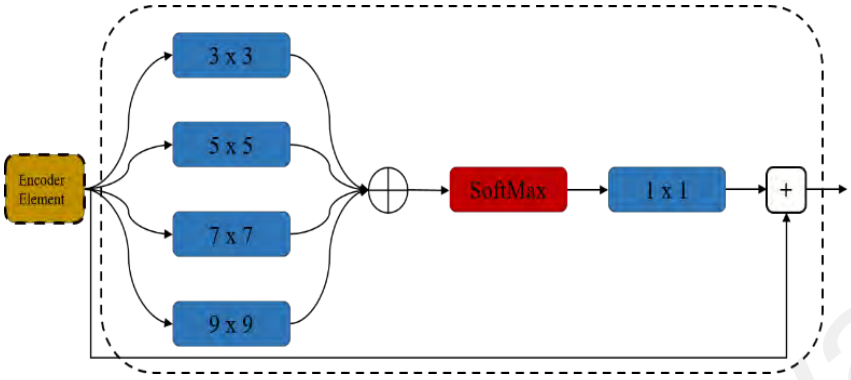

Figure 3.16: Additive attention gate

3.4.5.4 Activation Function

The activation function is a node between input and output that acts as a non-linear transformation function, removes the non-linearities from the input feature maps, and generates new feature maps to be inserted into the next layer. These non-linear functions have derivatives that update the weights in backpropagation with optimal value for each neuron in deep neural networks. In our proposed crack segmentation network, we used ReLU (Rectified Linear Unit) activation function which is applied after every convolutional operation in this architecture, as shown in Figure 3.17.

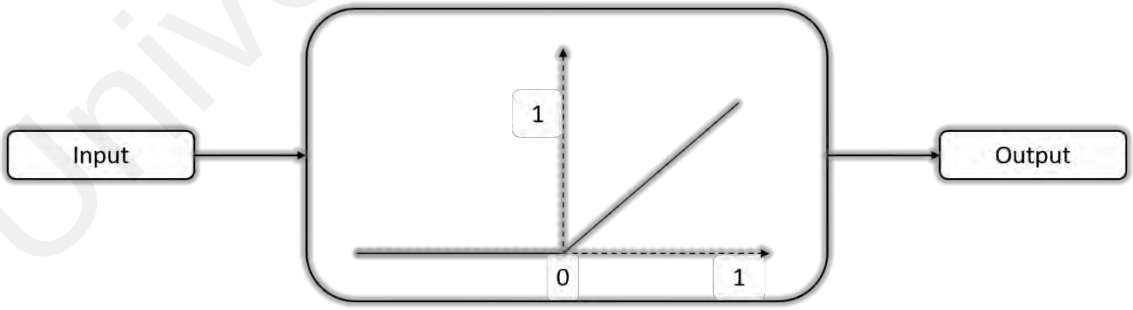

Figure 3.17: ReLU activation Function

ReLU is the most common and widely used activation function in DL image segmentation tasks. It has piecewise in nature and deactivates all the negative neurons by converting these values into zero and only activating those neurons which have positive values. Hence, only active neurons are processed which makes the learning process

computationally efficient and provides a much faster computing rate. The ReLU activation function is defined as

$$ReLU = f(x_i) = \begin{cases} 0, & x_i < 0 \\ x_i, & x_i \geq 0 \end{cases} \tag{3.4}$$

and the derivative of the ReLU function is defined as

$$ReLU' = f'(x_i) = \begin{cases} 0, & x_i < 0 \\ 1, & x_i \geq 0 \end{cases} \tag{3.5}$$

3.4.5.5 Batch Normalization

A deep neural network has a complex structure for processing the image data due to a large number of layers and every layer updates the weights through backpropagation from output to input. This makes the training of neural networks a complicated process as each layer changes its input with updated weights during training. Therefore, the input of each layer is required to normalize (internal covariate shift) for each mini-batch training. Therefore, batch normalization is used to coordinate the updated weights of the respective layer in the network which increases the training process and less number of epochs are needed for convergence (G. Chen et al., 2019).

In this work, the Batch-Normalization layer (Ioffe & Szegedy, 2015) is deployed after every convolutional layer in the crack segmentation network. It normalizes the feature maps from the previous layer (convolutional layer) and each channel of the feature map is normalized by subtracting the mean value and then the result is divided by the standard deviation using Equation (3.6).

$$BN = \frac{x_i - \mu_{mb}}{\sqrt{\sigma_{mb}^2 + \epsilon}} \tag{3.6}$$

where x_i is input features, μ_{mb} is the mean of mini-batch, σ_{mb}^2 is the variance of the mini-batch, and ϵ is a numerical factor for stability. The batch normalization layer also shifts

and scales the output value through Equation (3.7) from BN to BN_{ss}, allow for the possibility that inputs with zero mean and unit variance are not optimal for the next layer.

$$BN_{ss} = \gamma BN + \beta \tag{3.7}$$

where γ is the scaling factor and β is the shifting factor, which is learned along with original model parameters and stored in the trained network information.

3.5 Proposed Local Weighting Factor

The crack dataset is a typical example of an unbalanced class problem; the images contain a disproportionate number of non-crack pixels and crack pixels. The difference between the number of training samples of two classes i.e., crack and non-crack affects the training process. The class which has the majority of training elements has more influence on the training process than the class which has fewer training elements. The trained network becomes biased and tends to classify the pixels as of the majority class. This variation in pixel numbers between two classes leads to the false prediction of a crack pixel in the testing process and the cumulative loss overwhelms the final loss. In (X. Zhang et al., 2019), the weighted cross-entropy loss function is used which applied median frequency class weights in the loss function. These weights either have a constant value or are calculated from the overall dataset which is not the best fit because every image has a different proportion of crack and non-crack pixels.

For our experiment, a Multi Structure Crack image (MSCI) dataset is used which has a large variation (88.6% non-crack pixels and 11.4% crack pixels) between crack pixels and non-crack pixels. To resolve the imbalanced data issue, a local weighting factor is designed which generates and assigns weights through inverse frequency, according to the ratio between crack and non-crack pixels from each image. The cross-entropy loss function $L(x)$ in Equation (3.8), is used as an objective function to update the network weights. The local weighting factor is integrated with the loss function to minimize the

class imbalanced issue. The local weighting factor is generated by calculating the number of pixels in each class. The number of pixels determines the class frequency of that image using Equation (3.9).

$$L(x) = -\mathcal{L}(x)\sum_{i=1}^{k}\left(t_i(x)\log\left(p_i(x)\right)\right) \tag{3.8}$$

$$\mathcal{L}(x) = \left[\left(\frac{x_1 + x_2}{x_1}\right), \left(\frac{x_1 + x_2}{x_2}\right)\right] \tag{3.9}$$

where x is the input image, k is the total number of pixels in the image ($k = x_1 + x_2$), x_1 is the number of crack pixels, x_2 is the number of non-crack pixels, $\mathcal{L}(x)$ is a local weighting factor which is the ratio of the total number of pixels ($x_1 + x_2$) to the number of pixels in each class in that image. The $\mathcal{L}(x)$ factor produces the higher weights for crack pixels because they are in minority and lower weights for non-crack pixels which are higher in number. The inverse frequency of crack pixels is used as the weights with crack pixels to balance the class difference and the inverse frequency of non-crack pixels is used as the weight with non-crack pixels to balance the class difference. The higher weights of crack pixels increase the chance of crack pixel detection, and the lower weights of non-crack pixels balance the chance of non-crack pixels detection. The $\mathcal{L}(x)$ balances both classes in that image through an appropriate local weighting factor.

3.6 Proposed Difference Transform Map

In semantic segmentation, the probability estimation of each pixel determines the class of object it belongs. The boundary pixels are considered the hardest pixels to predict because they possess both crack and non-crack features. The pixel intensity at the boundary becomes darker if it is a non-crack pixel which tends towards the intensity of crack pixels and the intensity becomes lighter if it is a crack pixel which tends towards the intensity of non-crack pixels. In crack pixels segmentation, there is no specific factor that differentiates or marks the boundary between crack pixels and non-crack pixels. Therefore, it is hard to determine the boundary pixels as crack and non-crack. The

labeling of crack images requires special attention to label crack pixels and non-crack pixels on the boundary as shown in Figure 3.18. The left image is the original crack image and the upper right image is the boundary image, where pixels from the crack region also have pixels with light intensities, and pixels from the non-crack region also have pixels with dark intensities.

Figure 3.18: Boundary pixels and dark intensity pixels

On the other hand, the boundary pixels are not the only pixels that are hard to predict in crack pixel classification. Pixels belonging to spot, stain, or any other region with dark intensities are also sensitive and hard to predict correctly. The pixel from these regions has similar features to crack regions. In this work, a sensitivity weighted difference transformed map $\mathfrak{D}_d(x)$ is proposed to incorporate with the loss function in Equation (3.10). The $\mathfrak{D}_d(x)$ not only emphasized on boundary pixels but other than boundary pixels are also considered.

$$L(x) = -\sum_{i=1}^{k} \mathfrak{D}_d(x) \left(\left(\frac{x_1 + x_2}{x_1} \right) * \left(t_i(x) \log\left(p_i(x)\right) \right) + \left(\frac{x_1 + x_2}{x_2} \right) * \left((1 - t_i(x)) \log\left(1 - p_i(x)\right) \right) \right) \tag{3.10}$$

$$\mathfrak{D}_d(x) = \mathfrak{D}(x) \text{ x } \mathcal{L}(x) \tag{3.11}$$

$$\mathfrak{D}(x) = \begin{cases} D\left(t_i(x), p_i(x)\right) = 1 \\ D\left(t_i(x), p_i(x)\right) = -1 \\ D\left(t_i(x), p_i(x)\right) = 0 \end{cases} \tag{3.12}$$

$$D\left(t_i(x), p_i(x)\right) = \left(t_i(x) - p_i(x)\right) \tag{3.13}$$

where $\mathfrak{D}_d(x)$ is a difference transform map. $\mathfrak{D}(x)$ determines the misclassified pixel from the targeted label $t_i(x)$ and predicted label $\hat{p}_i(x)$. The $\mathfrak{D}(x)$ produces three different determinant outputs. The first output is based on $D\left(t_i(x), \hat{p}_i(x)\right) > 0$, where the difference between the targeted label $t_i(x)$ and predicted label $\hat{p}_i(x)$ is greater than zero i.e. 1, which determines that the network was required to predict the pixel as a crack but the network predicted the pixel as a non-crack pixel which is the wrong prediction. The second output is based on $D\left(t_i(x), \hat{p}_i(x)\right) < 0$, where the difference between the targeted label $t_i(x)$ and predicted label $\hat{p}_i(x)$ is less than zero i.e. -1, which determines that the network predicted the pixel as a crack pixel but it is a non-crack pixel which is the wrong prediction again. The third output is based on $D\left(t_i(x), \hat{p}_i(x)\right) = 0$ where the difference between the targeted label $t_i(x)$ and predicted label $\hat{p}_i(x)$ is zero, which determines that the network predicted the pixels correctly whether it is a crack pixel or a non-crack pixel. The output of the first and second cases needs attention for accurate prediction. These pixels are weighted or penalized with $\mathcal{L}(x)$ by exploring the co-occurrence of two different classes of pixels in each image.

The higher weights generated by $\mathcal{L}(x)$ factor is used to weight all those pixels which come under the $D\left(t_i(x), \hat{p}_i(x)\right) > 0$ output. This increases the chance of the occurrence of crack pixels. Similarly, the lower weights generated by $\mathcal{L}(x)$ factor are used to weight all those pixels which come under the $D\left(t_i(x), \hat{p}_i(x)\right) < 0$ output. This decreases the chance of occurrence of non-crack pixels.

3.7 Proposed Crack Measurement

Crack characteristics such as length, width, and orientation are useful indicators to analyze the severity of the crack and the strength of the structure. These indicators help to determine the condition of exposure so that appropriate action can be carried out to ensure safety. Several image processing-based methods such as morphological operations, median axis algorithm, and Euclidean methods have been proposed to measure the length, width, and area of the crack in the given image. These methods shrink the crack into its skeleton which does not provide the precise measurement of crack characteristics.

The crack characteristic measurement proposed in this work is based on the combination of several image processing techniques. The output segmented mask generated by the trained network is first passed through crack connected pixels algorithm that removes the group of less than 50 connected pixels from the mask. This cleans the image and removes all noisy and irrelevant regions from the mask. The mask contains only two types of pixels i.e., crack and non-crack pixels, all the crack pixels are summed up to determine the area of the crack. The crack orientation is determined by scanning the mask pixel by pixel both row-wise and column-wise. This provides the number of pixels in each row and column that provides the width, length, and orientation of the crack in a given segmented mask. Figure 3.19 shows the procedure of crack characteristics measurement in terms of area, length, width, and orientation.

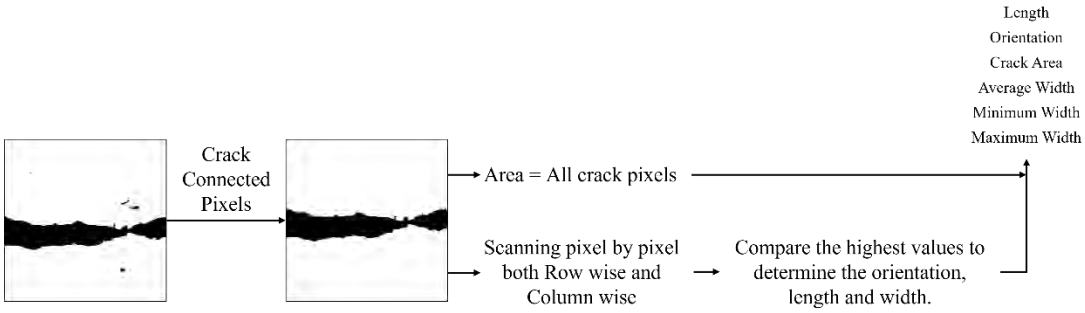

Figure 3.19: Crack characteristics measurement

3.7.1 Noise Removal

The output segmented mask generated by the trained network does not have 100% accuracy in determining the crack pixels. Few non-crack pixels are segmented as crack pixels which may cause an error in crack characteristics measurement. Therefore, before measuring the characteristics, the segmented mask is fast filtered to remove the regions which have less than 50 connected pixels (crack pixels). The value is optimized to 50 by analyzing the true cracks in the given test images. To remove the irrelevant crack pixels a connected crack pixel algorithm is designed to determine and remove these pixels.

3.7.1.1 Connected Crack Pixels

The segmented mask consists of two types of pixels, i.e., crack and non-crack pixels. These pixels are represented with black (crack pixel) and white (non-crack pixel) colors. Therefore, they have a unique identity that is used to determine and differentiate between crack and non-crack pixels. The connected components are a set of pixels that are connected and possess the same identity. The connected crack pixels algorithm starts from the top left corner, find and stores the pixel identity and compare it with all its neighboring pixel. If the unique identifier is the same, the value is updated in that direction and the process continues until the last pixel or change in the unique pixel identity. Then the process began in the next neighboring pixel and continue the same process in all possible directions. In this way, each cluster value is compared with a pre-defined constant value and if the value is less than or equal to 50, the pixels change their identity else the pixels keep their unique identity. The process is repeated up to the last pixel of the segmented mask.

3.7.2 Crack Orientation

After removing the irrelevant cluster of crack pixels, pixel-by-pixel scanning is performed both row-wise and column-wise. The scanning process keeps the record of the highest number of crack pixels in each row and column. The highest value in either direction will determine the orientation of the crack in that image. Equation (3.14) is used to compare and determine the orientation of the crack.

$$CO = \begin{cases} P(C)_{x-max} > P(C)_{y-max} & Vertical\ Crack \\ P(C)_{y-max} > P(C)_{x-max} & Horizontal\ Crack \end{cases} \quad (3.14)$$

where CO represents crack orientation, $P(C)_{x-max}$ is the maximum number of crack pixels in a column and $P(C)_{y-max}$ is a maximum number of crack pixels in a row. If the number of crack pixels in a column is higher than the number of crack pixels in a row then the orientation of the crack is vertical and if the number of crack pixels in a row is higher than the number of crack pixels in a column, than the orientation of crack is horizontal.

Algorithm 1: Orientation Algorithm

1 **Started Scanning the Segmented Mask:** *P(x, y)*
2 x ← 1
3 **while** x = 256 do
4 Count crack pixels in each row $P(C)_x$
5 Store number of crack pixels
6 **end**
7 y ← 1
8 **while** y = 256 do
9 Count crack pixels in each column $P(C)_y$
10 Store number of crack pixels
11 **end**
12 Compare the number of pixels in each row and column: $P(C)_x == P(C)_y$
13 **Crack Orientation:** The highest number of crack pixels in any row or column defines the orientation

if

$P(C)_{x-max} > P(C)_{y-max}$

Vertical Crack

else

$P(C)_{y-max} > P(C)_{x-max}$

Horizontal Crack

end

3.7.3 Crack Length

The length of the crack is determined by scanning the segmented mask in the direction of the orientation of the crack. It is the highest number of crack pixels in any row/column. If the crack is determined as a vertical crack, then the highest number of crack pixels in a column is crack length. If the crack is determined as a horizontal crack, then the highest number of crack pixels in a row is crack length.

Algorithm 2: Length Measurement Algorithm

Thinning the Crack: *P(C)*

if

C is even

Take the first-pixel value

Keep track of this value

else

C is odd

Take the central value

Keep track of this central value

end

Length: Trace all these pixels and count to get the length

3.7.4 Crack Width

The crack width is determined by scanning the segmented mask in the opposite direction of the orientation of the crack. It is the highest number of crack pixels in any row/column. If the crack is determined as a vertical crack, then the highest number of crack pixels in a row is the maximum crack width. If the crack is determined as a horizontal crack, then the highest number of crack pixels in a column is the maximum crack width. The width of the crack is not constant throughout the crack and keeps changing. Therefore, in this work, three different values for crack width i.e., maximum width, minimum width, and average width are measured to provide the closest value for further consideration.

3.7.5 Crack Area

The segmented mask contains either crack or non-crack pixels. Therefore, after removing the irrelevant and noisy pixels from the segmented mask, all the crack pixels are summed up using Equation (3.15) to determine the area occupied by crack pixels in the given mask. The crack area (CA) is,

$$CA = \sum P(C) \tag{3.15}$$

where CA represents the crack area and $P(C)$ is all the crack pixels in the mask.

CHAPTER 4: RESULTS

4.1 Introduction

In this section, the experimental results of crack pixels segmentation of the proposed crack segmentation network architecture, local weighting factor, sensitivity transform map, and crack characteristics measurement are presented. The dataset used in this experiment has two imbalanced classes, i.e. crack and non-crack. To evaluate the segmentation network, the dataset is portioned into 80% (400 images) for training, 10% (50 images) for validation, and 10% (50 images) for testing, respectively. These images are trained over five different neural networks including FCN (J. Long et al., 2015), U-Net (Ronneberger et al., 2015), SegNet (Badrinarayanan et al., 2017), DeepLabv3+ (L.-C. Chen et al., 2018), and the proposed Crack Segmentation Network (CSN). Each network is trained with six different loss functions, i.e. Cross-Entropy loss, Weighted Cross-Entropy loss, Dice loss, Tversky loss, Focal loss, and the novel local weighted loss function. To evaluate the performance of networks and loss functions, the most common evaluation measures for semantic segmentation or pixel classification are mean accuracy along with crack pixels and non-crack pixels accuracy, Precision, Recall, F1-score, and Jaccard distance (Taha & Hanbury, 2015), which are collectively explained with advantages and disadvantages in (Csurka et al., 2013).

4.2 Network Training Configurations

Crack segmentation neural network for generating segmentation mask of crack and non-crack pixels is trained using a desktop with Intel® Core (TM) i5-6400 CPU @ 2.7 GHz Processor, 16GB RAM, and NVIDIA GeForce GTX 730 with 2GB RAM GPU. The network is initialized using Glorot Initializer (Xavier Glorot & Bengio, 2010) which brings significantly faster convergence. The weights are randomly initialized from a uniform distribution with zero mean and variance. The Stochastic Gradient Descent

(SGD) is implemented to update each weight parameter in every iteration. To avoid the local minima trapped during the training process and increase the convergence rate (Bottou & Bousquet, 2009), the momentum approach is adopted. During the training process, stochastic gradient descent (SGD) (Rumelhart et al., 1986) with 0.9 momentum is chosen as an optimizer. The initial learning rate, weight decay, and L_2 regularization is set to 10^{-3}, 10^{-4}, and 10^{-4}, respectively. Due to memory constraints, the batch size is set to 2 with 10 shuffled epochs for training. The image dataset is portioned into 80% for training, 10% for validation, and 10% for testing, respectively. The training progress is analyzed with a different set of hyperparameters, and the best-validated model configuration is selected. The implementation of the FCN, U-Net, SegNet, DeepLabv3+, and proposed crack segmentation network architectures are based on Neural Network Toolbox in MATLAB R2020a.

4.3 Metrics

To analyze the proposed crack segmentation network with a local weighting factor and difference map, the proposed network is implemented using six different loss functions i.e., Cross-Entropy loss, Weighted Cross-Entropy loss, Dice loss, Tversky loss, Focal loss, and the novel local weighted loss function. The FCN, U-Net, SegNet, and DeepLabv3+ architectures are also implemented using these six different loss functions. In (Csurka et al., 2013), the most common evaluation measures for semantic segmentation or pixel classification are collectively explained with advantages and disadvantages. To evaluate the performance of the proposed crack segmentation network, loss function and difference map on the new dataset, mean accuracy along with crack pixels and non-crack pixels accuracy, precision, recall, boundary-F1 (BF), Dice similarity coefficient, Jaccard distance, and Hausdorff distance are used. These metrics are used to measure the accuracy of crack classification and segmentation.

All the metrics are derived from four basic elements, as shown in Figure 4.1. The elements are true positive (TP), true negative (TN), false positive (FP), and false-negative (FN). Here true positive (TP) is the input is a crack pixel and output is also classified as a crack pixel. True negative (TN) is the input is a non-crack pixel and the output is also classified as a non-crack pixel. False-positive (FP) is the input as a non-crack pixel and the output is misclassified as a crack pixel. False-negative (FN) is the input as crack pixel false and predicted as a non-crack pixel.

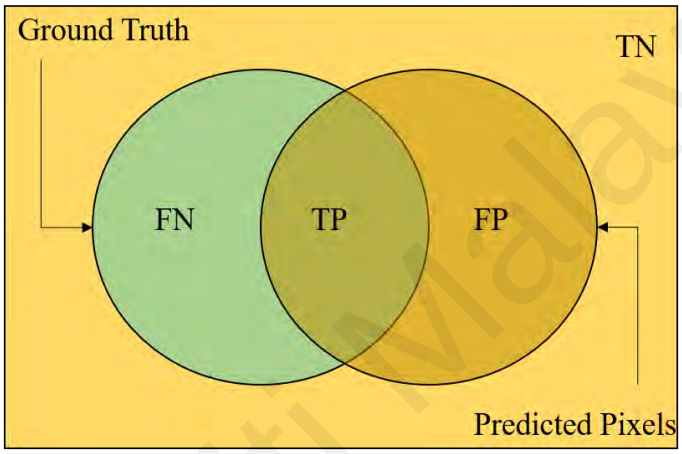

Figure 4.1: Metric elements

4.3.1 Accuracy

The metric accuracy determines the percentage of correctly identified pixels in each class. The accuracy of the individual class is determined by the ratio of correctly identified pixels in that class to the total number of pixels in that class.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.1}$$

There are two other terms related to accuracy; global accuracy which is the percentage of correctly predicted pixels and average accuracy which is the prediction of correct pixels in overall classes.

In our work, we mainly focus on the accuracy of crack and non-crack pixels. Due to the class imbalance issue, the majority class accuracy is more dominant than minority

class accuracy. Hence, the proposed method has balanced the class difference and achieved a high percentage of accuracy in both classes.

4.3.2 Intersection over Union

Intersection over Union (IoU) is also one of the parameters used to evaluate semantic segmentation models. It is the ratio between the intersection of actual crack pixels and predicted crack pixels to the union of actual crack pixels and predicted crack pixels as shown in Figure 4.1. IoU is defined in Equation (4.2).

$$\text{Intersection over Union} = \frac{Overlapping\ area}{Combined\ area} = \frac{TP}{TP + FP + FN} \tag{4.2}$$

This metric is also known as the Jaccard similarity coefficient. It is also used to measure the statistical accuracy, which penalizes FP.

4.3.3 Dice similarity coefficient

The Dice coefficient-based metric is one of the most used metrics for the evaluation of semantic segmentation. Its measurement is based on the overlapping comparison between the ground truth and the predicted segment.

$$\text{DICE} = \frac{2TP}{2TP + FP + FN} \tag{4.3}$$

4.3.4 Hausdorff distance

The Hausdorff distance (HD) between ground truth (GT) and predicted (P) segment is defined by

$$\text{HD}(\text{GT}, \text{P}) = \max\left(h(GT, P), h(P, GT)\right) \tag{4.4}$$

where $h(GT, P)$ is direct Huasdraoff distance presented by

$$\text{h}(\text{GT}, \text{P}) = \max_{g \in GT} \min_{p \in P} \|g - p\| \tag{4.5}$$

4.3.5 Precision

Precision is the ratio of true positive (TP) pixels and all the predicted positive pixels.

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{True\ Positive}{Actual\ Result} \tag{4.6}$$

4.3.6 Recall

A recall is the measure of correctly identified true positive pixels. It is the ratio of positive pixels in the ground truth to positive pixels identified in the segmented image.

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{True\ Positive}{Predicted\ Result} \tag{4.7}$$

4.3.7 F1-Measure

F1-Score or F1-measure is a model evaluation metric that summarizes the model performance. It combines both precision and recall. F1-measure is a weighted harmonic mean, calculated through recall and precision value.

$$\text{F1} - \text{Score} = \frac{2 * TP}{2 * TP + FP + FN} = \frac{Precision * Recall}{Precision + Recall} \tag{4.8}$$

4.4 Results

The performance of the proposed network is evaluated in terms of pixel-level accuracy. Both numerical and visual results are used to evaluate the crack segmentation network.

4.4.1 Numerical Results

The performance of the proposed CSN network is evaluated in terms of pixel-level accuracy. Different loss functions are integrated with the proposed network and pixel-level accuracy is calculated. The predicted results of test images demonstrated that the local weighting factor and difference transform map performed well on all CNN's and our proposed network also performed well with other loss functions than other networks.

Table 4.1 summarizes the results of the FCN network architecture with the integration of all the loss functions used in this research work. The network does not detect the crack pixels accurately. From the results of CE, Dice, Tversky, and focal loss function it can be observed that the network does not tackle the imbalanced issue and the accuracy of non-crack pixels are higher than the accuracy of crack pixels. The WCE has obtained better accuracy for both classes while the LWF-DTM has achieved the highest crack pixel accuracy than non-crack pixels. The LWF-DTM achieved 96.79% crack pixel accuracy, which is the highest crack pixel accuracy among all the loss functions. Dice loss achieved 96.80% non-crack pixel accuracy which is the highest non-crack pixel accuracy among all the loss functions. The highest mean accuracy is 93.82% obtained by the LWF-DTM loss function.

Table 4.1: Results of FCN architecture

Loss Functions	Crack Pixels Accuracy	Non-Crack Pixels Accuracy	Accuracy
CE	0.7848	0.9677	0.8762
WCE	0.9650	0.8848	0.9249
Dice	0.7060	**0.9680**	0.8370
Tversky	0.6574	0.8583	0.7579
FL	0.7533	0.8644	0.8089
LWF-DTM	**0.9679**	0.9086	**0.9382**

Table 4.2 summarizes the results of the U-Net architecture with the integration of all the loss functions used in this research work. The network is capable to detect both crack and non-crack pixels with more than 90% accuracy. From the results of CE, Dice, and focal loss function it can be observed that the network does not tackle the imbalanced issue and the accuracy of non-crack pixels are higher than the accuracy of crack pixels.

The WCE, Tversky, and LWF-DTM have obtained higher crack pixel accuracy than non-crack pixels. The LWF-DTM achieved 98.08% crack pixel accuracy, which is the highest crack pixel accuracy among all the loss functions. CE loss achieved 99.24% non-crack pixel accuracy which is the highest non-crack pixel accuracy among all the loss functions, but the crack pixel accuracy is 90.41% which is the lowest among all the loss functions. The highest mean accuracy is 96.96% obtained by the LWF-DTM loss function.

Table 4.2: Results of U-Net architecture

Loss Functions	Crack Pixels Accuracy	Non-Crack Pixels Accuracy	Accuracy
CE	0.9041	**0.9924**	0.9482
WCE	0.9745	0.9530	0.9637
Dice	0.9516	0.9777	0.9646
Tversky	0.9787	0.9567	0.9677
FL	0.9307	0.9819	0.9563
LWF-DTM	**0.9808**	0.9584	**0.9696**

Table 4.3 summarizes the results of the SegNet architecture with the integration of all the loss functions used in this research work. The SegNet does not perform well to identify the crack pixels accurately whereas the accuracy of non-crack pixels is more than 90%. From the results of CE, Dice, WCE, Tversky, and focal loss function it can be observed that the network does not tackle the imbalanced issue and the accuracy of non-crack pixels is higher than the accuracy of crack pixels. The LWF-DTM has obtained higher crack pixel accuracy than non-crack pixels. The LWF-DTM achieved 96.64% crack pixel accuracy, which is the highest crack pixel accuracy among all the loss functions. CE loss achieved 98.38% non-crack pixel accuracy which is the highest non-crack pixel accuracy among all the loss functions, but the crack pixel accuracy is 44.01%

which is the lowest among all the loss functions. The highest mean accuracy is 95.86% obtained by the LWF-DTM loss function.

Table 4.3: Results of SegNet architecture

Loss Functions	Crack Pixels Accuracy	Non-Crack Pixels Accuracy	Accuracy
CE	0.4401	**0.9838**	0.7120
WCE	0.7132	0.9791	0.8462
Dice	0.6306	0.9670	0.7988
Tversky	0.7399	0.9737	0.8568
FL	0.7125	0.9304	0.8215
LWF-DTM	**0.9664**	0.9509	**0.9586**

Table 4.4 summarizes the results of the DeepLabv3+ architecture with the integration of all the loss functions used in this research work. The network is not able to tackle the class imbalanced issue as the accuracy of non-crack pixels is higher than crack pixels. Both classes have achieved very low accuracy, less than 80%. From the results of CE, Dice, and focal loss function it can be observed that the network does not tackle the imbalanced issue and the accuracy of non-crack pixels is higher than the accuracy of crack pixels. The WCE, Tversky, and LWF-DTM have obtained higher crack pixel accuracy than non-crack pixels. The LWF-DTM achieved 77.41% crack pixel accuracy, which is the highest crack pixel accuracy among all the loss functions. CE loss achieved 77.43% non-crack pixel accuracy which is the highest non-crack pixel accuracy among all the loss functions with 69.71% crack pixel accuracy. The highest mean accuracy is 76.27% obtained by the LWF-DTM loss function.

Table 4.5 summarizes the results of the CSN architecture with the integration of all the loss functions used in this research work. The network tackles the class imbalanced issue

Table 4.4: Results of DeepLab3v+ architecture

Loss Functions	Crack Pixels Accuracy	Non-Crack Pixels Accuracy	Accuracy
CE	0.6971	**0.7743**	0.7357
WCE	0.7651	0.7562	0.7607
Dice	0.6899	0.7496	0.7197
Tversky	0.6609	0.6330	0.6470
FL	0.6653	0.7309	0.6981
LWF-DTM	**0.7741**	0.7514	**0.7627**

very well along with LWF-DTM as the accuracy of non-crack pixels is nearly equal to crack pixel accuracy. Both classes have achieved very high accuracy, with more than 95% average accuracy. From the results of the CE loss function, it can be observed that the crack pixel accuracy is lower than non-crack pixels whereas for the other loss functions the accuracy is either higher or nearly equal for both classes. The LWF-DTM achieved 98.61% crack pixel accuracy, which is the highest crack pixel accuracy among all the loss functions, and 98.35% non-crack pixel accuracy which is also the highest non-crack pixel accuracy among all the loss functions. The highest mean accuracy is 98.48% obtained by the LWF-DTM loss function.

Table 4.5: Results of the proposed crack Segmentation network

Loss Functions	Crack Pixels Accuracy	Non-Crack Pixels Accuracy	Mean Accuracy
CE	0.9358	0.9798	0.9578
WCE	0.9841	0.9725	0.9783
Dice	0.9673	0.9811	0.9742
Tversky	0.9717	0.9830	0.9773
FL	0.9790	0.9796	0.9793
LWF-DTM	**0.9861**	**0.9835**	**0.9848**

The proposed CSN has achieved more than 98% accuracy for both classes. The individual accuracy of crack and non-crack pixels is almost equal which indicates that the network is capable to predict the crack pixels with equal accuracy along with non-crack pixels by balancing the class imbalance issue.

4.4.2 Visual Results

The visualization of segmented crack pixels is shown in Figure 4.2, Figure 4.3, Figure 4.4, Figure 4.5, and Figure 4.6. The results of FCN architecture along with different loss functions are shown in Figure 4.2, the results of U-Net architecture along with different loss functions are shown in Figure 4.3, the results of Seg-Net architecture along with different loss functions are shown in Figure 4.4, the results of DeepLabv3+ architecture along with different loss functions are shown in Figure 4.5, and the results of proposed CSN architecture along with different loss functions are shown in Figure 4.6.

For the presentation, five images are randomly selected among the 50 test images. The first column contains original RGB images followed by respective ground truth masks in the second column. From column three to column eight the visual segmented results of the corresponding test images, trained by using different loss functions are shown. The observation from the segmented output mask predicted by FCN architecture with different loss functions shown in Figure 4.2, shows that the network could not be able to predict the crack pixels accurately and several non-crack pixels are around the crack region are also segmented as crack pixels. The segmented crack region by the FCN architecture is very thick and non-continuous. The network could not identify the minor cracks as shown in columns 1 and 5 of Figure 4.2. All the loss functions provided the rough and thick track of the crack region; therefore, the accuracy of the crack pixel is very low as compared to non-crack pixels for the majority of the loss functions. From the

numerical result and visual segmented output, it can be observed that the FCN architecture is not a suitable option for crack pixel segmentation of imbalanced data sets.

Figure 4.3 shows the performance of U-Net architecture with different loss functions for five randomly selected images. It can be observed that the U-Net architecture has predicted several non-crack pixels as crack pixels. These wrongly predicted pixels are either dark intensity pixels or boundary pixels. U-Net with CE loss function has less wrongly predicted crack pixels, but they also have lower crack pixel accuracy. Whereas WCE, Dice, Tversky, and Focal loss functions have high crack pixels accuracy but the predicted several non-crack pixels as crack pixels. They have a trade-off between precision and recall; therefore, they have a smaller number of wrongly predicted pixels whereas CE, WCE, and FL have a large number of wrongly predicted pixels and that is due to pixel-wise prediction. The results of the U-Net with LEF-DTM have a smaller number of wrongly predicted pixels and have a clearer view of pixel segmentation than other loss functions. The U-Net architecture has predicted the crack pixels more accurately as compared to FCN but due to class imbalanced dataset, the network could not be able to provide the clean view of segmented crack images.

The segmented images of SegNet architecture with different loss functions for five randomly selected images are shown in Figure 4.4. The segmented mask shows that the boundary pixels are wrongly predicted by the SegNet architecture. It has rough segmentation results on boundaries as compared to U-Net but has very few wrongly predicted pixels other than boundary pixels. The CE, WCE, Dice, Tversky, and Focal loss function has a thicker segmented crack region with wrongly predicted boundary pixels. The SegNet with WCE has wrongly predicted crack pixels at dark intensity areas. The other network has less wrongly predicted other than boundary pixels. The results of SegNet with LWF-DTM have better visual output as compared to the FCN and U-Net.

Overall, the SegNet architecture has predicted the crack pixels more accurately as compared to FCN and non-crack pixels more accurately as compared to U-Net. The network could not identify the minor cracks as shown in columns 1 and 5 of Figure 4.4.

The Deeplabv3+ network architecture performed very poorly for non-crack pixels as compared to crack pixels. Figure 4.5 shows the segmented output result of five randomly chosen images from the test image dataset and it can be observed that the segmented mask shows a gray background which is non-crack pixels predicted as crack pixels. The crack region also contains the wrongly predicted pixels, white pixels inside the crack region which appears like a net. For all loss functions, DeepLabv3+ has shown crack region tracing accurately but due to the network structure, the accuracy rate is very low for both crack and non-crack regions and is not a suitable architecture for the imbalanced crack dataset.

The proposed network performed better than FCN, U-Net, SegNet, and DeepLabv3+ network architecture. The visual results from Figure 4.6 show that the output segmentation masks have a larger similarity with ground truth and have lesser wrongly predicted pixels for both crack and non-crack regions. The CSN accurately predicts the boundary and dark intensity areas with more than 95% mean accuracy for all loss functions. The CE, Dice, and Tversky performed well but could not predict the minor cracks accurately whereas WCE, Focal, and LWF-DTM predict the tracing of minor cracks. The network along with all the loss functions achieved more than 97% non-crack pixel accuracy and 93% for crack pixel accuracy. The architecture is proven to be another best-fit network architecture for crack segmentation tasks for imbalanced datasets. The performance of all loss functions shows better accuracy, among them LWF-DTM shows the best performance for tackling class imbalanced effect by achieving 98.61% crack pixel accuracy and 98.35% non-crack pixel accuracy with 98.48% mean accuracy.

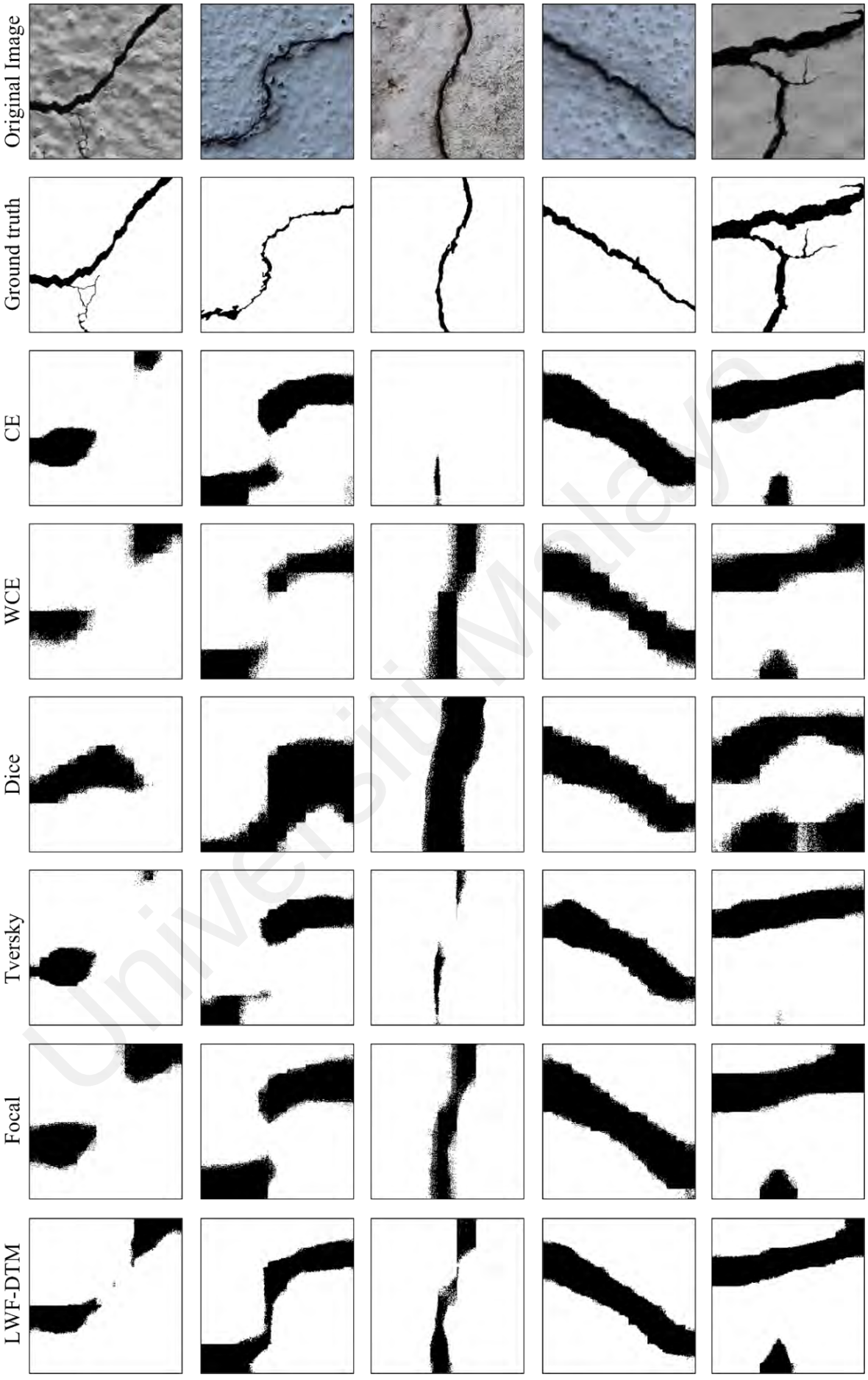

Figure 4.2: Crack Detection by FCN Architecture using different loss functions

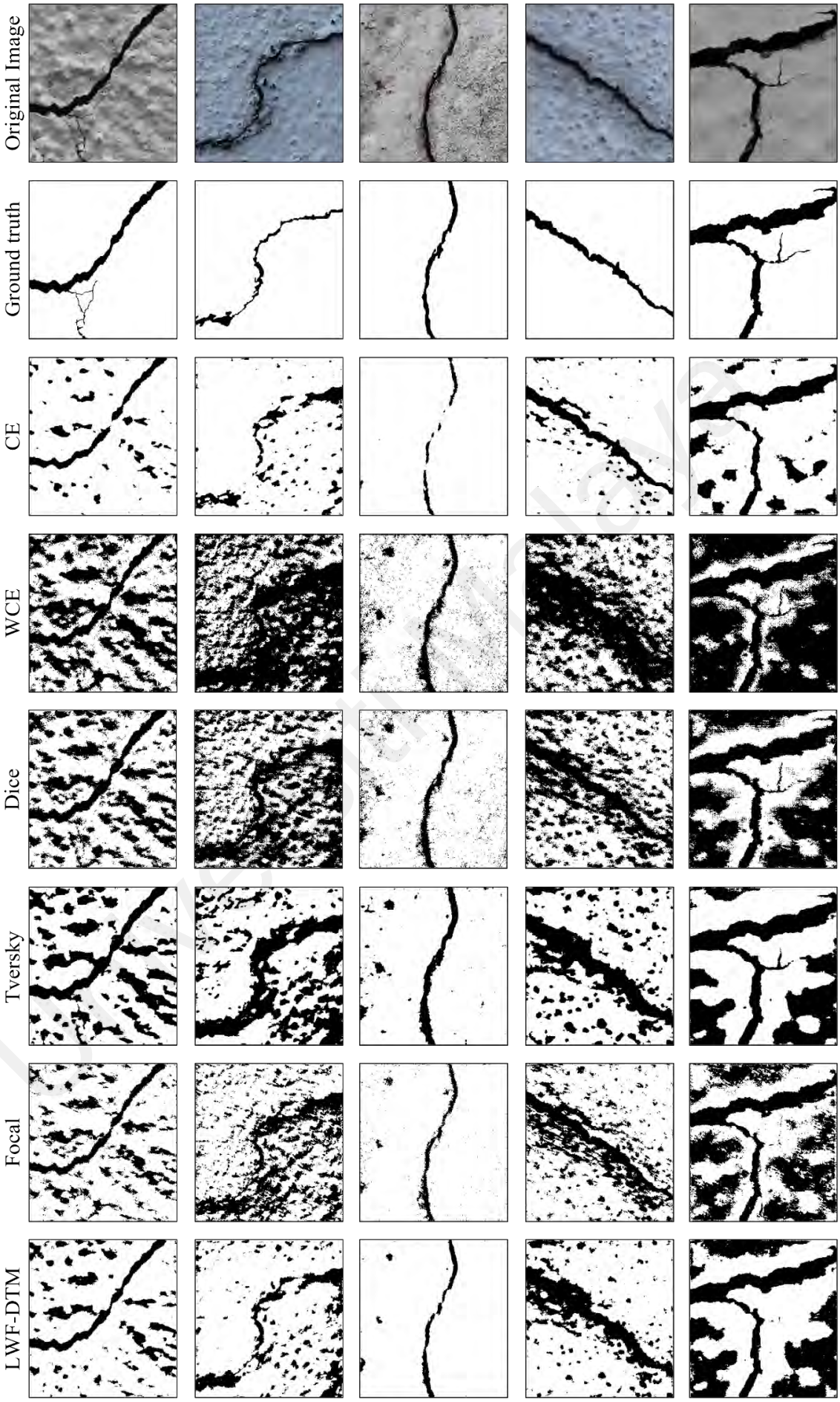

Figure 4.3: Crack Detection by U-Net Architecture using different loss functions

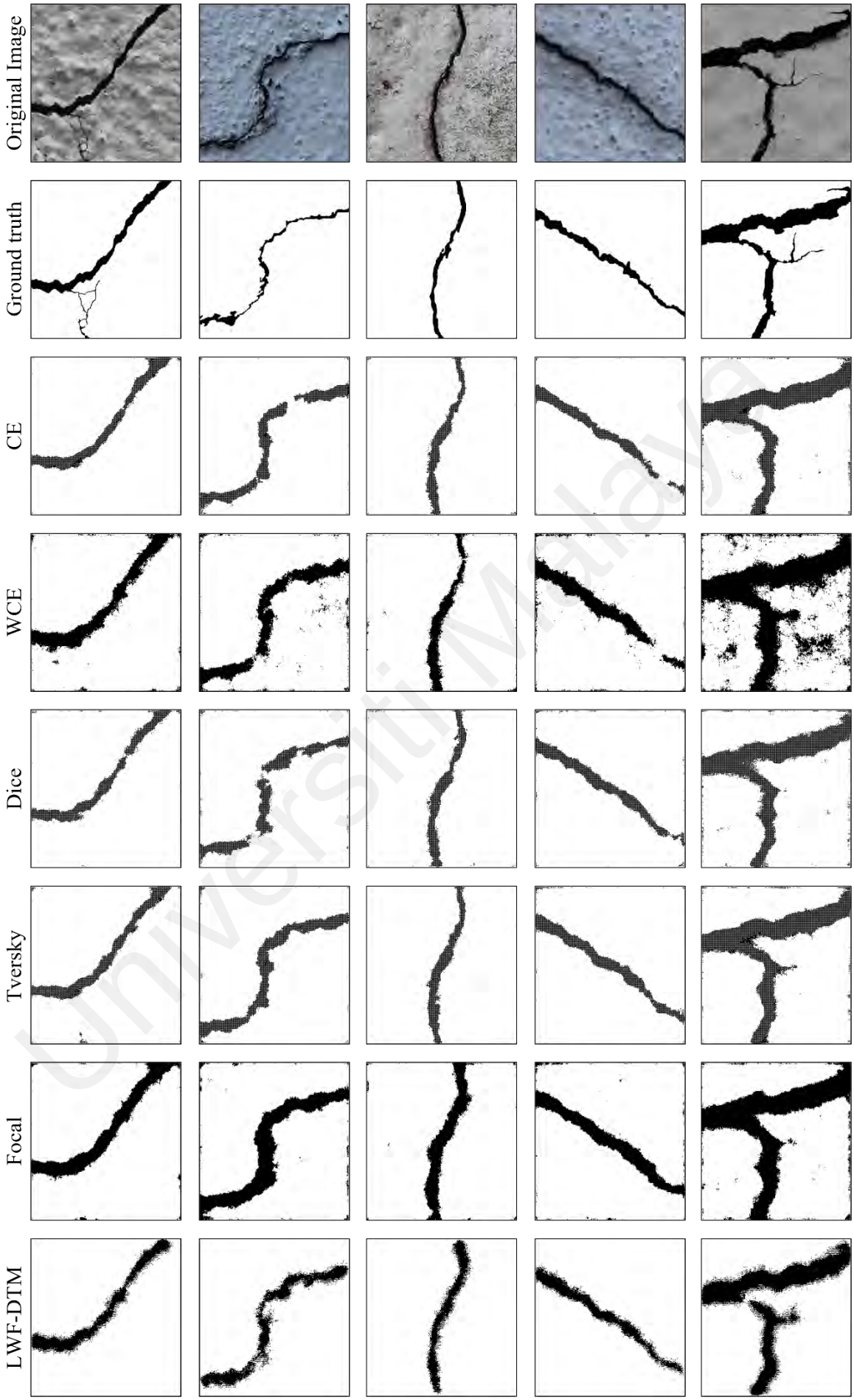

Figure 4.4: Crack detection by SegNet architecture using different loss functions

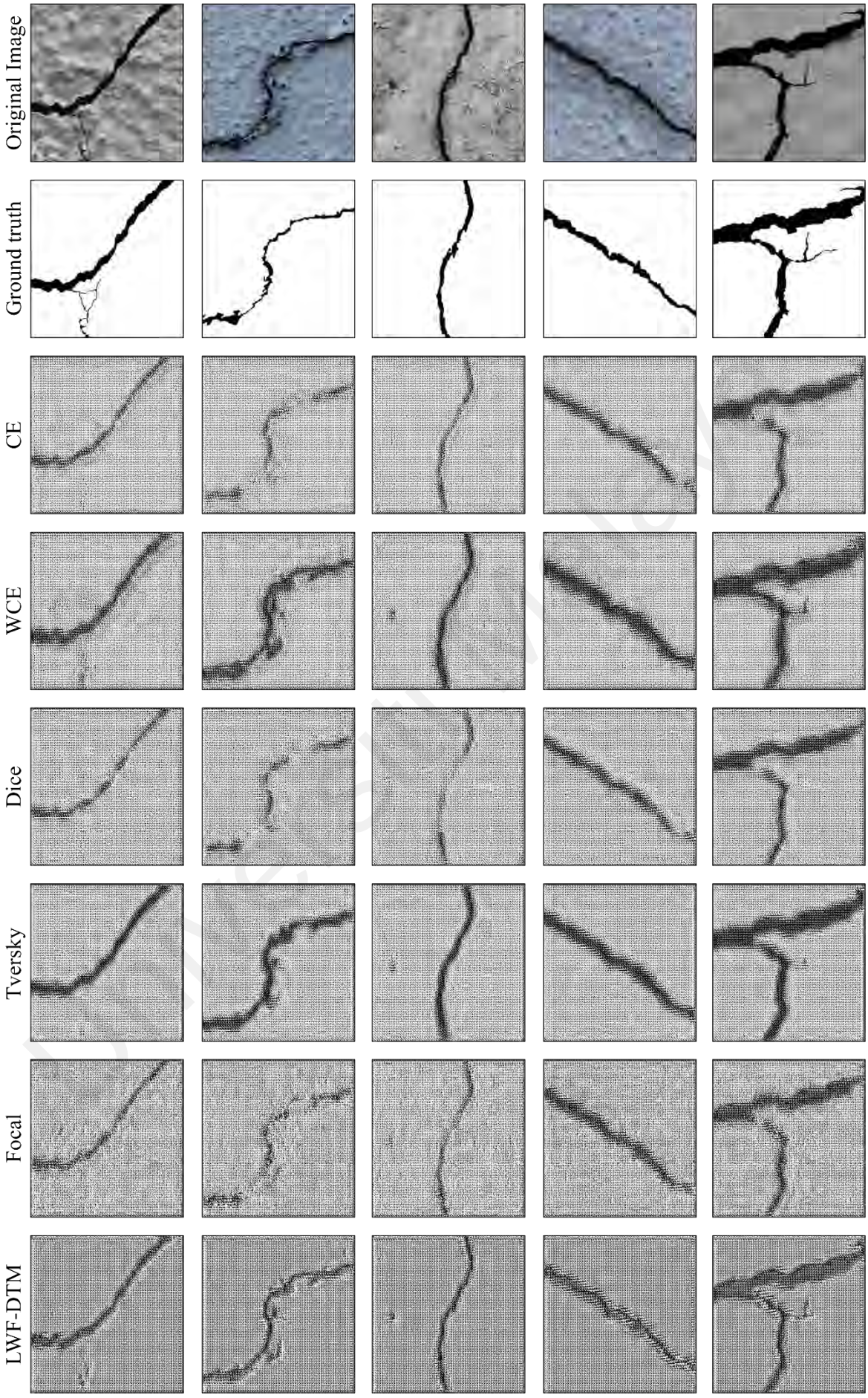

Figure 4.5: Crack detection by DeepLabv3+ Architecture using different loss functions

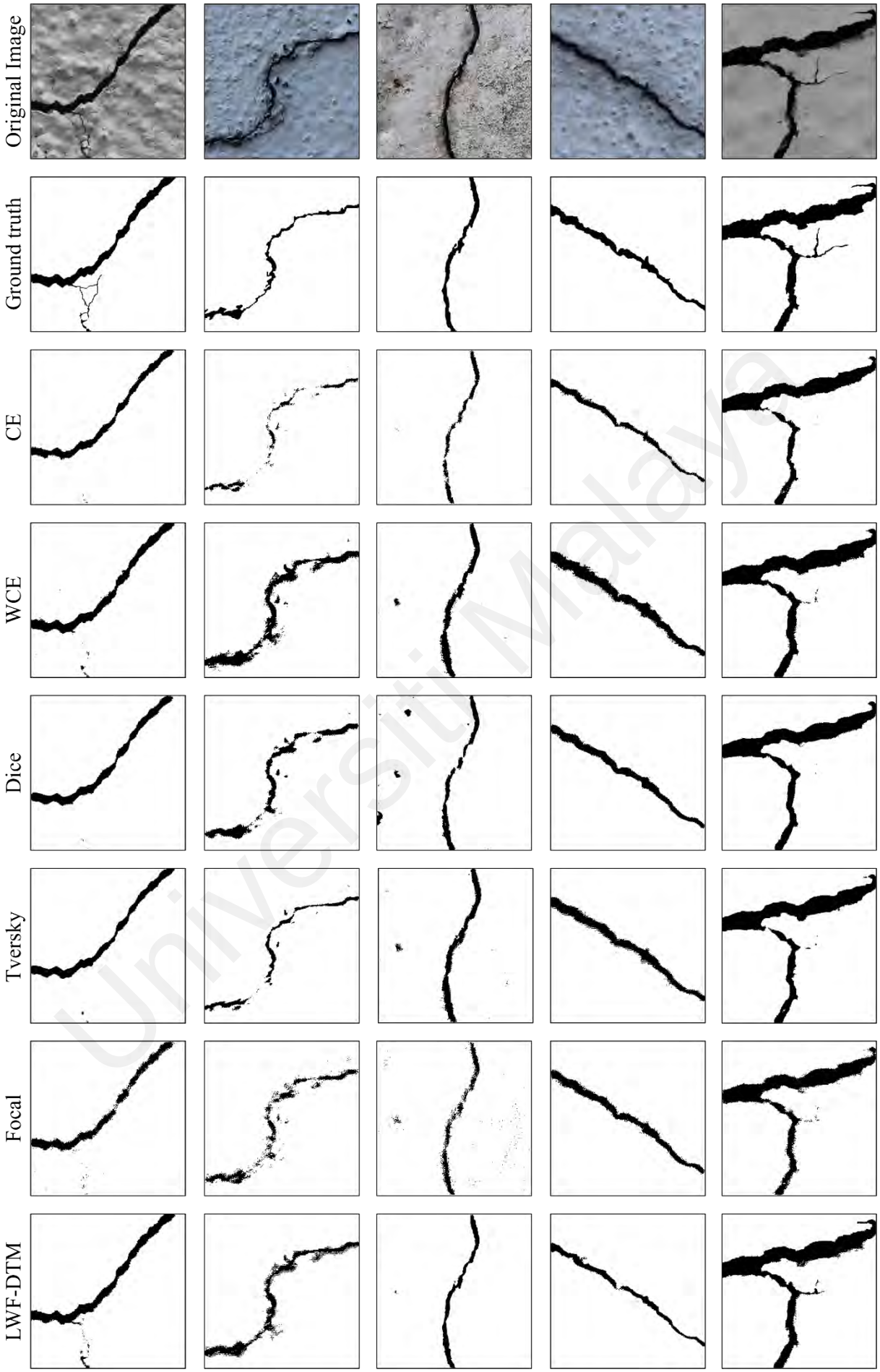

Figure 4.6: Crack detection by Proposed CSN Architecture using different loss functions

4.4.3 Crack Characteristics Measurement

The crack characteristics measurement process is shown in Figure 4.7. The segmented mask is cleaned after processing through the trained network. The cleaned image is then processed through an image processing-based measurement technique to measure the orientation, length, width, and area of a crack region in the given image.

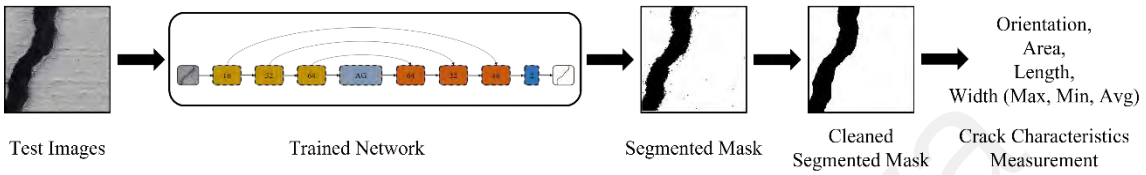

Figure 4.7: Crack measurement process

The results of crack characteristics measurement for images selected from the test dataset are presented in Figure 4.8. The image processing-based crack characteristics measurement algorithm measures the crack orientation, area, length, and width. The algorithm measure 3 values for crack width as crack width varies at each point on the crack, i.e. maximum width, average width, and minimum width. The accuracy of measuring the orientation of vertical and horizontal cracks is 100% whereas the accuracy of diagonal cracks is 85%. Figure 4.9 presents the result of crack orientation detection of the segmented mask. The result shows that diagonal cracks are the hardest to identify because most of the diagonal cracks are not fully diagonal, they are either vertical diagonal or horizontal diagonal as shown in the second and third rows of Figure 4.8. The system detects these types of cracks as vertical or horizontal cracks which leads to the low accuracy of crack orientation accuracy. The width of the crack is considered the most critical character to be measured. Therefore, to provide maximum information about the width of the crack in any given image, maximum width, minimum width, and average width. The length of the crack is determined by tracing the central pixel. Similarly, after cleaning the segmented mask from noise, all the pixels are summed up to provide the area of the crack. The calibration pixel and measurement are not set in this work as the

calibration depends on the distance between the capturing device and crack which differs from dataset to dataset.

In this work, the unit of length, width, and area are not specified. To set the unit of these characteristics, calibration is required. The distance between the capturing device and the crack is needed to accurately measure and specify the pixels per unit. Images in every data set are taken from a variable distance which cannot be used to calibrate and define the unit. Therefore, in this work, all the crack measurements are based on pixels rather than other measuring units. The unit can be set by calibrating the images or by measuring the distance between the crack and capturing device, so that, the number of pixels can be used to measure the length, width, and area according to their standard units.

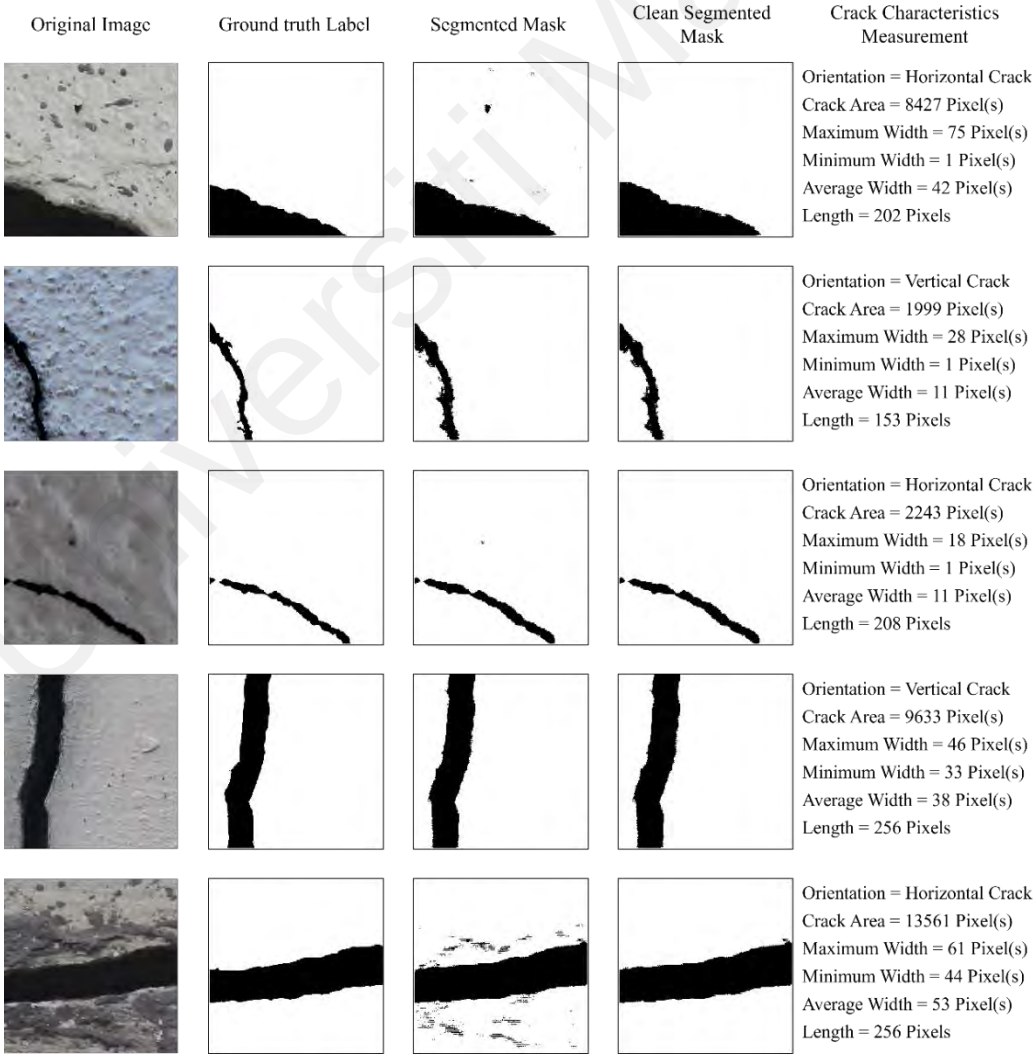

Figure 4.8: Crack characteristics measurement

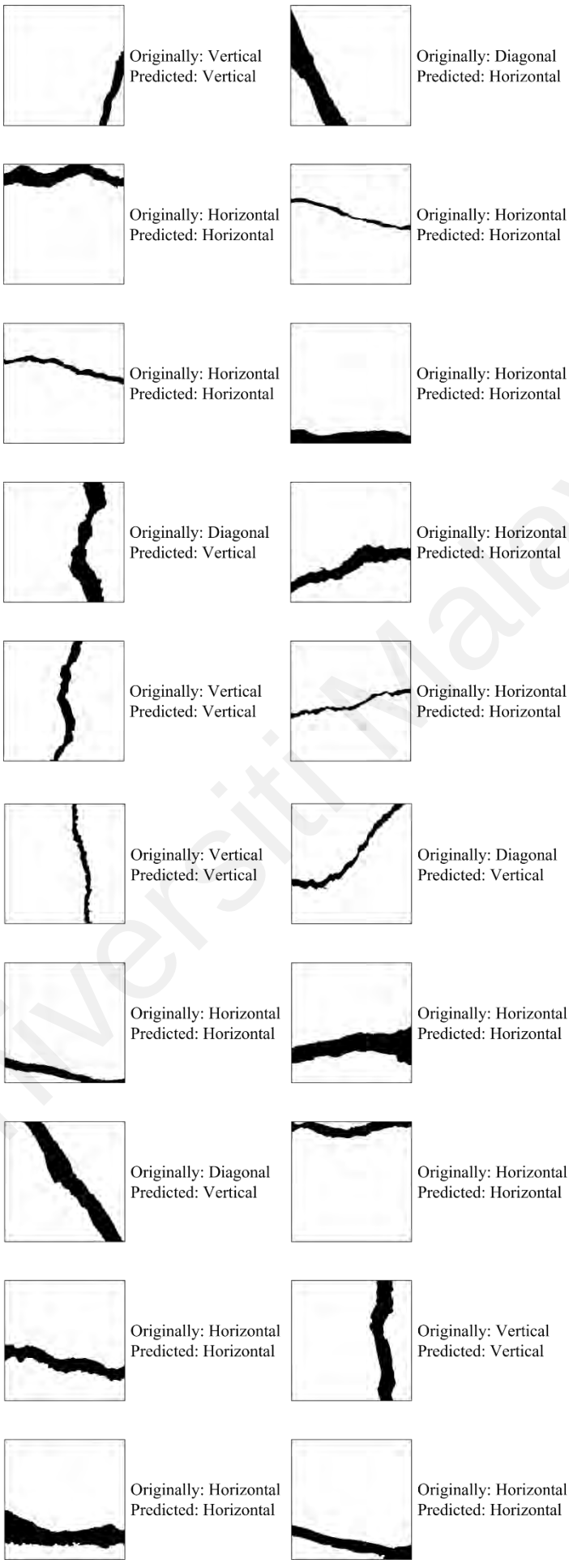

Figure 4.9: Crack Orientation measurement

CHAPTER 5: DISCUSSION

This chapter contains the interpretation of the results presented in chapter 4. The findings of the research are evaluated and compared with those of previous studies presented in the literature review. The purpose of this chapter is to discuss the findings and the outcomes of the research to the results that have been obtained.

5.1 Introduction

Analysis of the performance of the proposed CSN architecture and LWF-DTM is discussed in this section. The network architecture was designed to accurately classify the pixels with minimum computation complexity and less processing time. Furthermore, the integration of AAG uses a localization module to connect the encoder with the decoder and increases the network learning ability. The LWF-DTM increased the segmentation accuracy for both classes at the pixel level in an imbalanced condition. The effect of spot, stain, and confusion at boundary regions was minimized through LWF-DTM. Results demonstrated that the CSN architecture improves the segmentation accuracy and LWF-DTM equals the pixel-level accuracy of each class. A comparison of CSN with other network architectures and LWF-DTM with other loss functions is presented and discussed in the next section.

5.2 Network Comparison

All the network architectures implemented in this work are quantified utilizing crack and non-crack pixel accuracy, mean accuracy, precision, recall, F1-Score, Jaccard distance, number of learnable parameters, and training time. The most important performance evaluation metric is crack and non-crack pixel accuracy. The imbalanced dataset issue highly affects the network performance which causes the difference of accuracy in both classes.

For the evaluation of our proposed CSN architecture, the network is compared with four state-of-the-art semantic segmentation network architectures with different loss functions. The main objective was to show that the LWF-DTM as a loss function has better performance than other existing loss functions. Moreover, the proposed CSN network architecture with LWF-DTM has superior results as compared to other networks and loss functions.

From the experimental results, it is shown that FCN architecture could not handle the imbalanced dataset efficiently, the accuracy of non-crack pixels is very high for CE, Dice, Tversky, and Focal loss functions whereas the accuracy of crack pixels is very low. For WCE and LWF-DTM the accuracy of the non-crack pixel is lower as compared to crack pixels. The two-loss functions specifically add weighting factors to support the minority class. The WCE loss function uses a constant weighting value for all the images which is not the best fit, although the crack pixel accuracy is higher, the constant factor cannot help to increase the accuracy of every image in the dataset. The LWF-DTM uses the variable weighting factor to support the minority class by calculating the value for each respective image in the dataset. The FCN architecture has an up-sampling operation which tends to smooth the crack details and the minor crack are ignored as FCN often losses small details which have been presented in visual results.

The U-Net architecture has shown better performance for imbalanced datasets. The accuracy of non-crack pixels is very high for CE, Dice, and Focal loss functions whereas the crack pixel's accuracy is lower. For WCE, Tversky, and LWF-DTM the accuracy of the non-crack pixel is lower as compared to crack pixels. The LWF-DTM has achieved almost 97% average accuracy, the highest among the other loss functions. Although the U-Net architecture has high accuracy as compared to FCN architecture, it still cannot differentiate between dark and regions which are not cracked. From the visual result, the

U-Net architecture segmented the dark intensity pixels as crack pixels which ultimately increases the accuracy of crack pixels but on the other hand, the average accuracy remains below 97%. The middle layers of U-Net architecture experience low learning which causes the ignorance of abstract features. This causes the network to associate these dark intensity pixels with crack pixels.

The SegNet architecture performed better than the FCN but lower than the U-Net architecture. It only achieved higher accuracy of crack pixels using LWF-DTM while with CE, WCE, Dice, Tversky, and Focal loss function the accuracy of the non-crack pixel is much higher than crack pixels. The SegNet architecture achieved more than 95% accuracy for both crack and non-crack pixels and the LWF-DTM is the only loss function that achieved higher accuracy with SegNet. The architecture segmented the thicker cracks with acceptable accuracy but for smaller and fine cracks, the network was unable to identify them as crack pixels. The SegNet suspected inefficiency for thin and small cracks which exploit low-level and high-level features for the network.

The DeepLabv3+ network architecture tackled the imbalanced issue but could not able achieve the high mean accuracy. Although the network performed better against the class imbalanced dataset, individual accuracy of both crack and non- crack pixels remained below 78% which is the worst performance among other networks. The structure of DeepLabv3+ consists of hole convolutions with excessive expansion rates which affect the extraction of targeted features and the relation between local features of the large-scale target cannot be simulated. Therefore, from the visual result, it can also be verified that the features are not properly and completely extracted which causes the segmentation to degrade. The LWF-DTM with DeepLabv3+ can accurately classify the crack pixels with 77% accuracy, the highest among the other loss functions.

From Table 5.1, it can be observed that the accuracy of all the networks using LWF-DTM as a loss function has performed well in FCN, U-Net, SegNet, and DeepLavbv3+.

The proposed CSN network architecture is proven to be another best-fit network architecture for crack segmentation tasks for imbalanced datasets. The performance of FCN, U-Net, SegNet, and DeepLavbv3+ shows better accuracy by achieving 93.82%, 96.96%, 95.86%, and 76.27% mean accuracy respectively. The CSN architecture has achieved 98.48% mean accuracy, with LWF-DTM and showed the best performance for tackling class imbalanced effects. Similarly, among all the loss functions implemented in this work, the performance of LWF-DTM shows the best performance for tackling class imbalanced effects by achieving 98.48% mean accuracy for CSN architecture.

Table 5.1: Results of local weighting factor & difference transform map

Network Architecture	Crack Pixels Accuracy	Non-Crack Pixels Accuracy	Accuracy
FCN	0.9679	0.9086	0.9382
U-Net	0.9808	0.9584	0.9696
SegNet	0.9664	0.9509	0.9586
DeepLabv3+	0.7741	0.7514	0.7627
Proposed CSN	**0.9861**	**0.9835**	**0.9848**

5.3 Loss function Comparison

The performance analysis of the proposed LWF-DTM on different segmentation network architectures is discussed in this section. The performance is measured on several metrics and compared with other loss functions for an imbalanced dataset. The dataset used in this research work is highly imbalanced. The MSCI dataset consists of two classes, the crack pixels that belong to the minority class and non-crack pixels that belong to the majority class. The number of crack pixels is very less as compared to non-crack pixels which steer the network performance. The LWF-DTM proposed in this work

efficiently tackles the class imbalanced issue and accurately classifies the pixels equally for both classes.

Table 5.2, Table 5.3, Table 5.4, Table 5.5, Table 5.6, and Table 5.7 compare the results of cross-entropy loss, weighted cross-entropy loss, dice loss, Tversky loss, focal loss, and LWF-DTM by implementing on FCN, U-Net, SegNet, DeepLabv3+, and proposed crack segmentation network. To evaluate the segmentation accuracy of the proposed LWF-DTM for crack pixel segmentation, fifty crack images were selected for testing the trained networks. These images were evaluated on the Precision, Recall, F1-Score, Jaccard Distance, Training Time, and Learnable Parameters. Due to imbalanced data, the network becomes biased and tends to predict more non-crack pixels because they have a large number of training examples than crack pixels. Therefore, the comparison in the tables shows that crack pixel accuracy is always lower than non-crack pixel accuracy.

Furthermore, the results of each loss function along with different network architectures are analyzed. Each loss function has produced a large number of accurate non-crack pixels as compared to crack pixels, but the LWF-DTM has almost the same accuracy percentage for both types of pixels. On the other hand, the proposed network and loss function has a minimum difference between the accuracy of non-crack pixels and crack pixels as compared to other networks and loss functions.

The performance of the proposed CSN architecture and LWF-DTM is also evaluated with other network architectures and loss functions on other segmentation evaluation metrics such as Precision, Recall, F1-Score, and Jaccard Distance on unseen test images. These evaluation metrics are also important evaluation methods for semantic segmentation. The precision and recall are relevant terms and are fraction relevant instances among retrieved instances or that were retrieved, respectively. The F1-Score is the harmonic mean of recall and precision. Jaccard distance determines the percentage matching of ground truth and segmentation mask in a numeric scalar or a numeric vector.

The evaluated metrics result of all the networks architecture and loss functions are tabulated from Table 5.2 to Table 5.7.

Table 5.2: Comparison among CNN architectures using a CE loss function

Network Architecture	Precision	Recall	F1-score	Jaccard Distance
FCN	0.6805	0.7848	0.7289	0.5735
U-Net	**0.9126**	0.9041	**0.9083**	**0.8320**
SegNet	0.8010	0.4402	0.5681	0.3968
Deeplabv3+	0.2132	0.6971	0.3266	0.1951
Proposed-CSN	0.8081	**0.9357**	0.8792	0.7845

Table 5.2 shows the evaluation of the CE loss function for all the network architectures implemented in this research work. The CE loss function has acceptable performance for U-Net and proposed CSN architecture whereas for other networks either precision or recall value is low. Similarly, the evaluation of loss function on Jaccard distance is better for U-Net and CSN. The reason behind the poor performance of the CE loss function for crack segmentation for the imbalanced dataset is its averaging nature because the CE loss function evaluates each pixel prediction individually and averages them to get over all pixel predictions. This causes CE to equally sort both crack and non-crack pixels

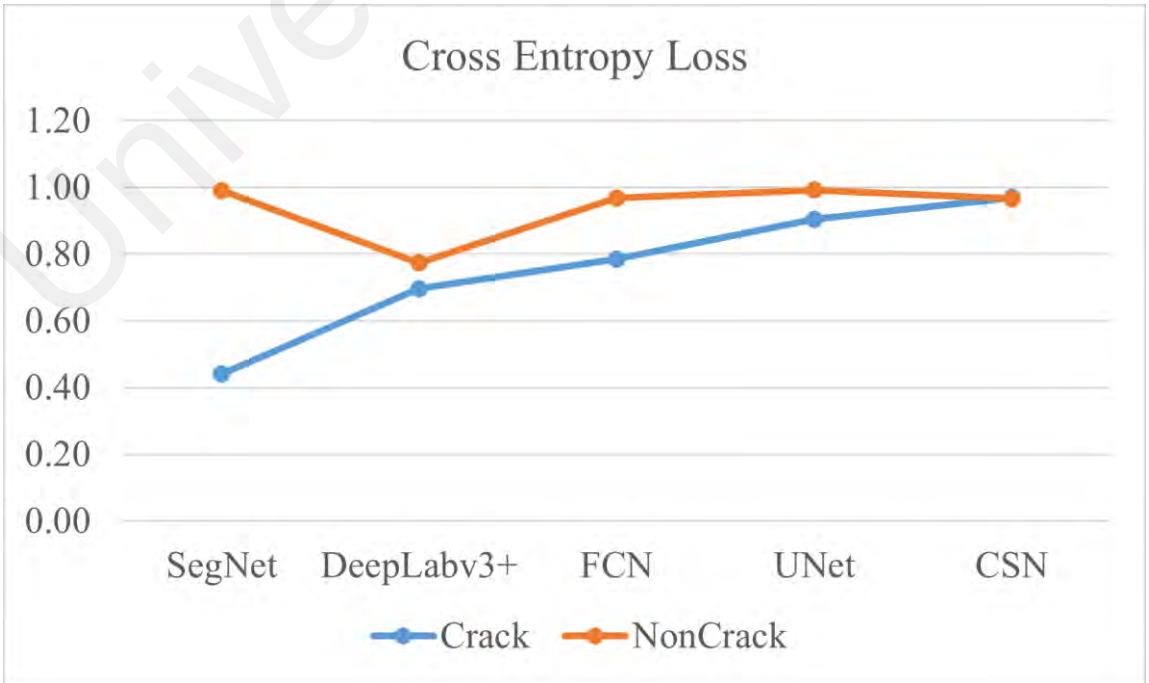

Figure 5.1: Evaluation of CE loss function

regardless of their imbalanced presence in the training process. The False-positive value of segmentation is very high which causes a reduction in precision value and similarly, the false negative value is also very high reflects in the degradation of recall value. The smaller region in the segmented mask which is not properly predicted by the CE loss function causes a low value of Jaccard distance. Figure 5.1 shows the pixel-level accuracy that the CE loss function has achieved more non-crack pixel accuracy which belongs to the majority class than crack pixels which belong to the minority class. The difference between the accuracy of crack and non-crack pixels varies due to network performance. Therefore, the CE loss function performed below average and is not a suitable choice for crack segmentation of an imbalanced dataset.

Table 5.3: Comparison among CNN architectures using a WCE loss function

Network Architecture	Precision	Recall	F1-score	Jaccard Distance
FCN	0.5341	0.9650	0.6876	0.5239
U-Net	0.5366	0.9935	0.6968	0.5347
SegNet	0.5230	**0.9945**	0.6855	0.5215
Deeplabv3+	0.2159	0.7651	0.3368	0.2025
Proposed-CSN	**0.6805**	0.9841	**0.8084**	**0.6785**

Table 5.3 presents the performance of the WCE loss function for all the network architectures implemented in this research work. The WCE loss function has acceptable performance for FCN, U-Net, SegNet, and proposed CSN architecture whereas for DeepLabv3+ networks the precision is very low and the recall value is below average. Similarly, the evaluation of loss function on F1-Score and Jaccard distance is very poor for DepLabv3+. The WCE loss function assigns extra weights to both classes to balance the difference in training elements. The recall value for WCE is very high which shows fewer false negative values are predicted by the networks using the WCE loss function, but it cannot manage to lower the false positive, which is due to extra weights. The

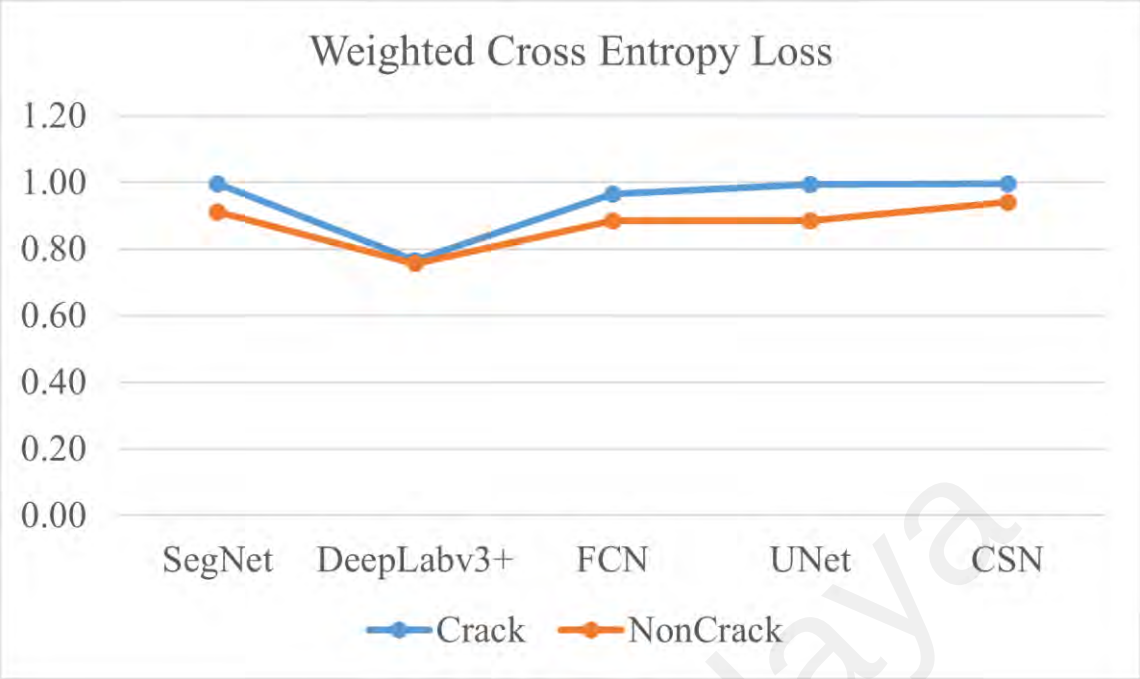

Figure 5.2: Evaluation of CE loss function

weights are constant numbers and assigned to every image regardless of the size of the crack present in the image. This causes WCE to favor both classes regardless of the amount of favor required. Figure 5.2 shows the pixel-level accuracy that the WCE loss function has achieved more crack pixel accuracy which belongs to the minority class than non-crack pixels which belong to the majority class because of extract weight assigned during the loss function. The difference between the accuracy of crack and non-crack pixels is small as compared to CE loss and varies due to network performance. It can also be observed that for CSN the accuracy is higher and the difference between crack and non-crack pixels accuracy is very small. Therefore, the WCE performed much better than CE and can be adopted as a loss function for crack segmentation of an imbalanced dataset.

Table 5.4: Comparison among CNN architectures using a Dice loss function

Network Architecture	Precision	Recall	F1-score	Jaccard Distance
FCN	0.6596	0.7060	0.6820	0.5174
U-Net	0.6891	0.9514	0.7993	0.6656
SegNet	0.7470	0.4413	0.5549	0.3840
Deeplabv3+	0.1947	0.6899	0.3036	0.1790
Proposed-CSN	**0.7857**	**0.9673**	**0.8705**	**0.7707**

Table 5.4 depicts the outcomes of the Dice loss function for all the network architectures implemented in this research work. The Dice loss function has average performance for FCN, U-Net, and SegNet whereas for the DeepLabv3+ network the loss function performed below average and for CSN the loss function performed above average. The precision is very low and the recall value is below average. Similarly, the evaluation of loss function on F1-Score and Jaccard distance is very poor for all the networks except CSN. The Dice loss function is designed to deal with an imbalanced dataset that favors one class and ignores the other. The recall values for Dice are higher which shows fewer false negative values are predicted by the networks using the Dice loss function, but it cannot manage to lower the false positive, which is due to ignorance of other classes. This causes Dice to favor only one target class and ignore the other class regardless of the importance of that class. Figure 5.3 shows the pixel-level accuracy that the Dice loss function has achieved more non-crack pixel accuracy which belongs to the majority class for SegNet, DeepLabv3+, and FCN than crack pixels which belongs to the minority class. The difference between the accuracy of crack and non-crack pixels is also high and varies due to network performance. The U-Net and CSN have performed better

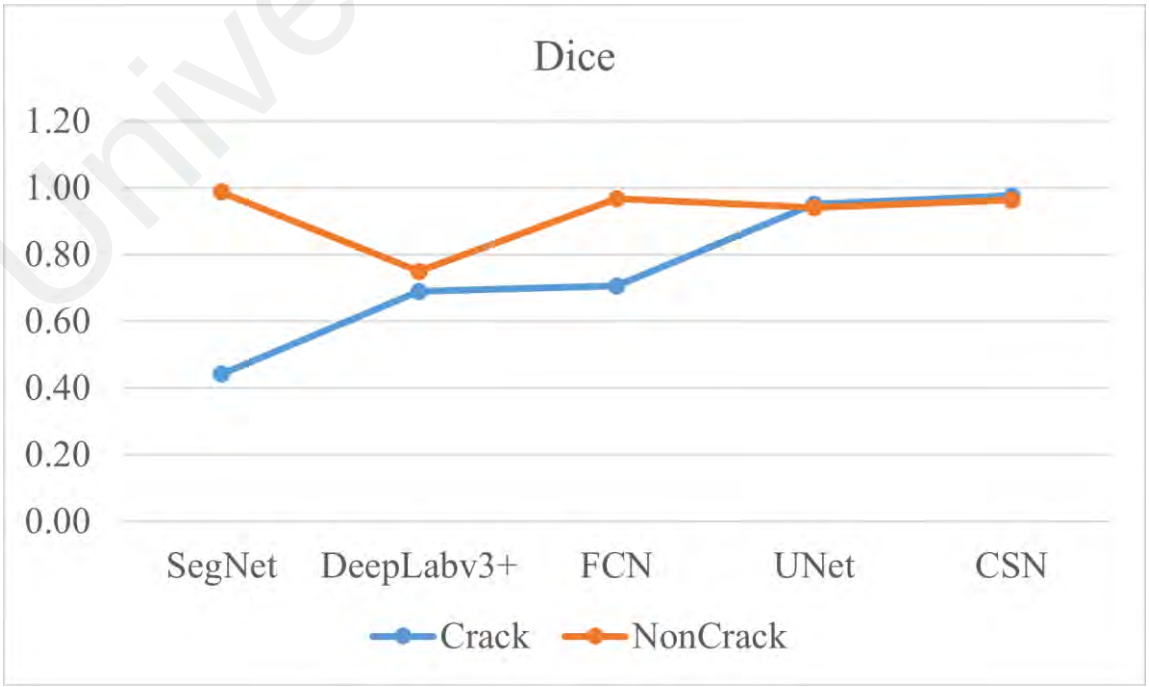

Figure 5.3: Evaluation of Dice loss function

and achieved higher crack pixel accuracy than a non-crack pixel and the difference between crack and non-crack pixel accuracy is also minimal. The Dice loss function performed much better than CE but lower than WCE and cannot be a good choice as a loss function for crack segmentation of imbalanced datasets for all the networks.

Table 5.5: Comparison among CNN architectures using a Tversky loss function

Network Architecture	Precision	Recall	F1-score	Jaccard Distance
FCN	0.5608	0.8453	0.6742	0.5086
U-Net	0.5752	**0.9986**	0.7299	0.5747
SegNet	0.7114	0.7399	0.7253	0.5690
Deeplabv3+	0.1364	0.6609	0.2262	0.1275
Proposed-CSN	**0.7799**	0.9717	**0.8704**	**0.7705**

Table 5.5 shows the results of the Tversky loss function for all the network architectures implemented in this research work. The Tversky loss function has average performance for FCN, U-Net, SegNet and whereas for the DeepLabv3+ network the loss function performed below average. The precision is very lower than the recall value that is because of false negative prediction. is below average. Similarly, the evaluation of loss function on F1-Score and Jaccard distance is average for all the networks except DeepLabv3+. The Tversky loss function is designed to add weights to FP (false positives)

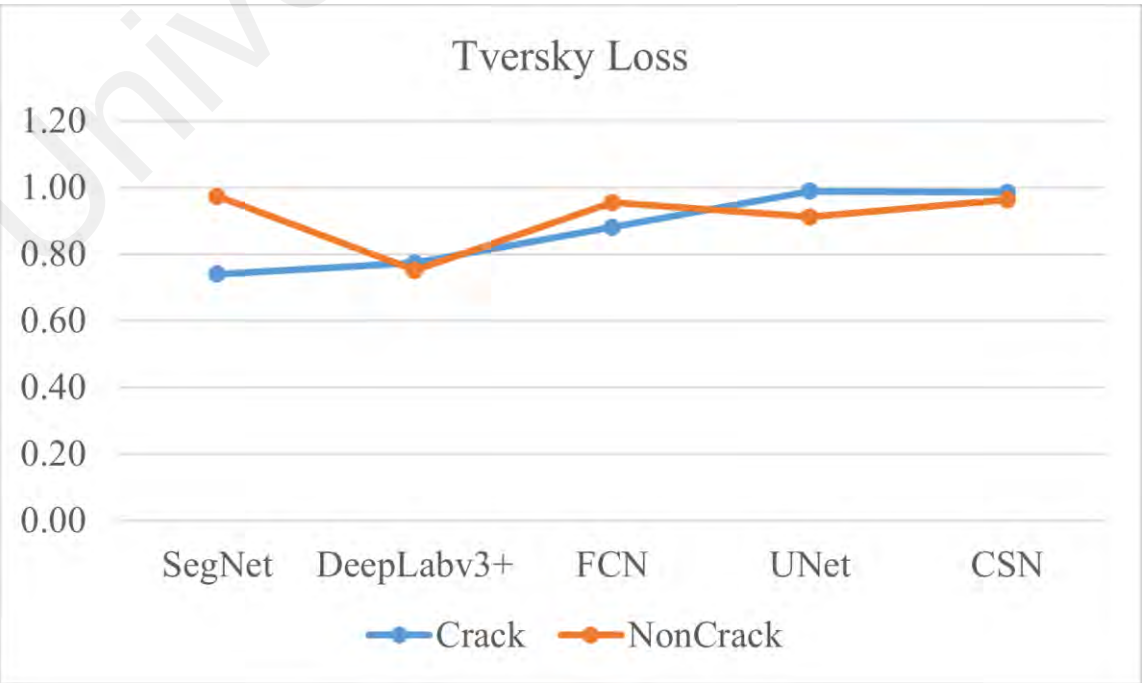

Figure 5.4: Evaluation of Tversky loss function

and FN (false negatives) to address the issue of imbalanced data in segmentation tasks through a trade-off between precision and recall. The recall values for Tversky are higher which shows fewer false negative values are predicted by the networks, but it cannot manage to lower the false positive. Figure 5.4 shows the pixel-level accuracy that the Tversky loss function has achieved more non-crack pixel accuracy which belongs to the majority class for SegNet and FCN than crack pixels which belong to the minority class. The difference between the accuracy of crack and non-crack pixels is also high and varies due to network performance. The DeepLabv3+, U-Net, and CSN have performed better and achieved higher crack pixel accuracy than a non-crack pixel and the difference between crack and non-crack pixel accuracy is also minimal. Tversky loss function performed similar to Dice loss and cannot be a good choice as a loss function for crack segmentation of imbalanced dataset for all the networks.

Table 5.6: Comparison among CNN architectures using a Focal loss function

Network Architecture	Precision	Recall	F1-score	Jaccard Distance
FCN	0.3276	0.7533	0.4567	0.2959
U-Net	**0.8183**	0.9307	**0.8709**	**0.7713**
SegNet	0.5628	0.7392	0.6390	0.4695
Deeplabv3+	0.1782	0.6653	0.2811	0.1636
Proposed-CSN	0.7866	**0.9790**	0.8650	0.7621

Table 5.6 presents the results of the Focal loss function for all the network architectures implemented in this research work. The Focal loss function has average performance for FCN, U-Net, SegNet and whereas for the DeepLabv3+ network the loss function performed below average. The precision is very lower than the recall value that is because the Focal loss function down-weights the negative class so that the positive class gets more attention from the network. Similarly, the evaluation of loss function on F1-Score and Jaccard distance is average for all the networks except DeepLabv3+. The Focal loss assists the network to focus on poorly well-trained pixels, therefore it performs well on an extreme imbalance dataset in segmentation tasks by focusing on the outliers and

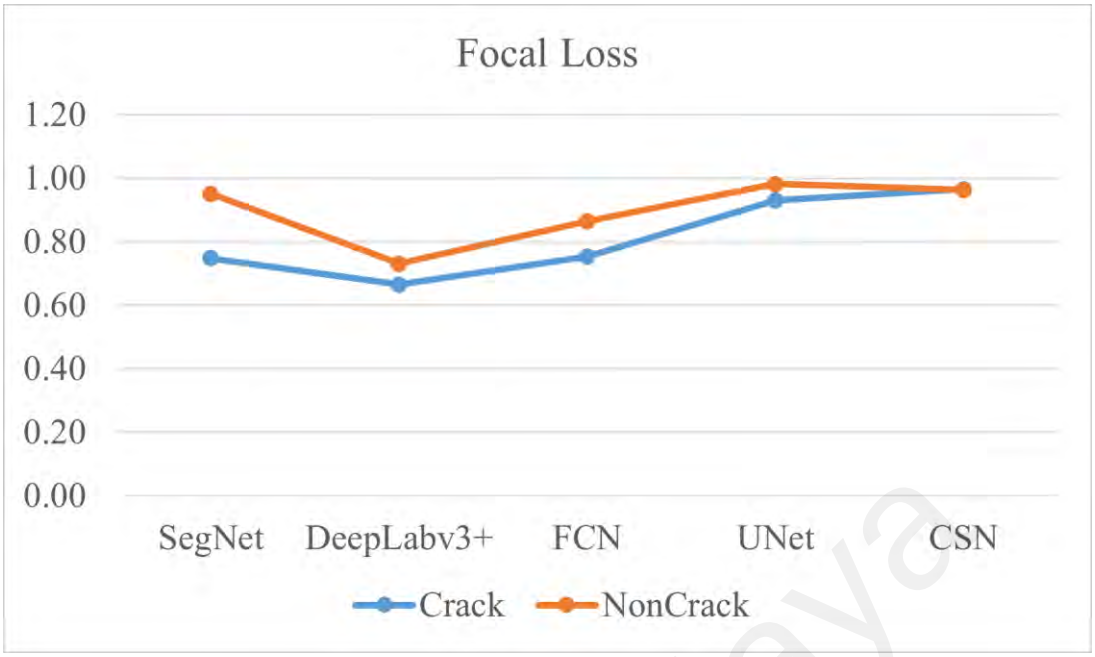

Figure 5.5: Evaluation of Focal loss function

misclassified samples by assigning higher weights to them. The recall values for the Focal loss function are higher which shows fewer false negative values are predicted by the networks, but it cannot manage to lower the false positive. Figure 5.5 shows the pixel-level accuracy that the Focal loss function has achieved more non-crack pixel accuracy which belongs to the majority class than crack pixels which belong to the minority class. The difference between the accuracy of crack and non-crack pixels is also high and varies due to network performance. The CSN has performed better and achieved higher crack pixel accuracy than a non-crack pixel and the difference between crack and non-crack pixel accuracy is also minimal. Focal loss function performed similar to Dice loss and Tversky loss function and cannot be a good choice as a loss function for crack segmentation of imbalanced dataset.

Table 5.7: Comparison among CNN architectures using an LWF-DTM

Network Architecture	Precision	Recall	F1-score	Jaccard Distance
FCN	0.5909	0.9113	0.7169	0.5588
U-Net	0.6708	0.9883	0.7992	0.6655
SegNet	0.6295	**0.9902**	0.7697	0.6256
Deeplabv3+	0.2145	0.7741	0.3360	0.2019
Proposed-CSN	**0.8402**	0.9861	**0.9073**	**0.8304**

Table 5.7 presents the performance of the LWF-DTM loss function for all the network architectures implemented in this research work. The LWF-DTM loss function has acceptable performance for FCN, U-Net, SegNet, and proposed CSN architecture whereas for DeepLabv3+ networks the precision is very low and the recall value is below average. Similarly, the evaluation of loss function on F1-Score and Jaccard distance is very poor for DepLabv3+. The LWF-DTM loss function assigns extra weights to both classes according to the requirement to balance the difference in training elements. The recall value for LWF-DTM is very high which shows fewer false negative values are predicted by the networks using the LWF-DTM loss function. Furthermore, the loss function somehow manages to lower the false positive, which is due to variable extra weights. The weights are fixed numbers and assigned to every image according to the size of the crack present in the image. This causes LWF-DTM to favor the low representative class and penalize the high representative class. Figure 5.6 shows the pixel-level accuracy that the LWF-DTM loss function has achieved more crack pixel accuracy which belongs to the minority class than non-crack pixels which belong to the majority class. The difference between the accuracy of crack and non-crack pixels is very small

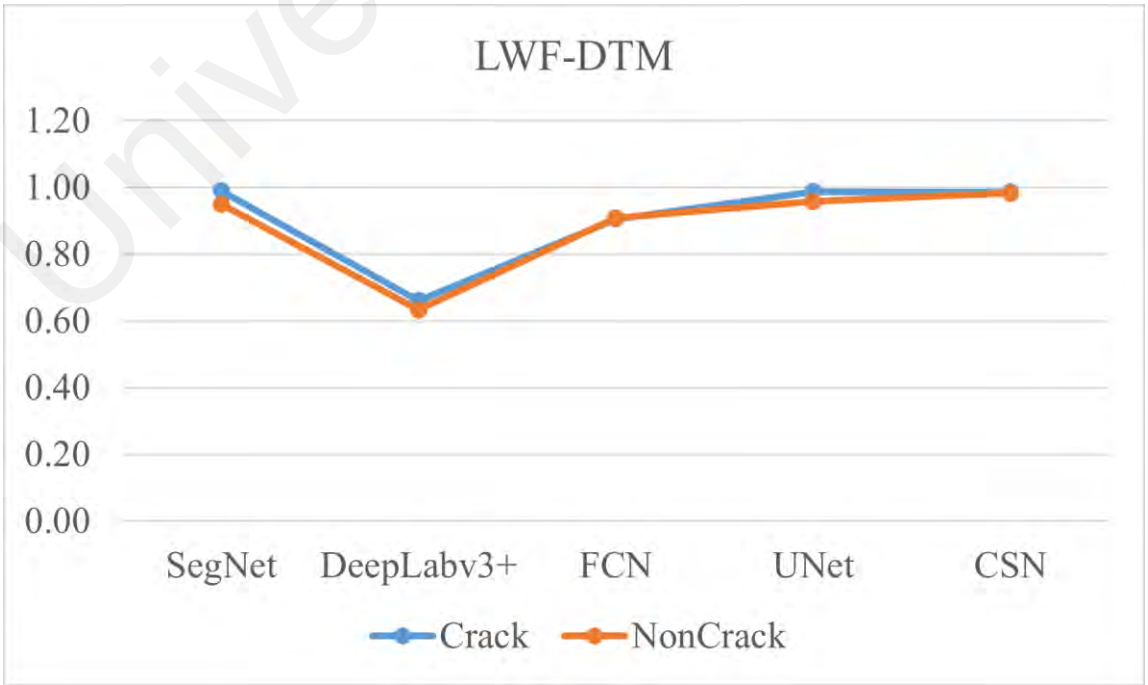

Figure 5.6: Evaluation of LWF-DTM loss function

and varies due to network performance. The CSN has performed better and achieved higher crack and non-crack pixels and the difference between crack and non-crack pixel accuracy is also minimal. Therefore, the LWF-DTM loss function performed better than other loss functions and is the best fit for crack segmentation of an imbalanced dataset.

5.4 Other Performance Measures

The network architectures implemented in this research work for crack segmentation have also been compared with each other for training time, the number of layers in each network, and learnable parameters. These performance evaluators are compared and presented in Table 5.8. All the network configurations such as hardware, software, initial weights, and other network parameters mentioned in Section 4.2 are set the same for each training.

Table 5.8: CNN architectures comparison

Network Architecture	Training Time	No. of Layers	Learnable Parameters
FCN	52 minutes	43	6.9 million
U-Net	177 minutes	58	31 million
SegNet	76 minutes	59	0.5 million
Deeplabv3+	33 minutes	100	20 million
Proposed-CSN	36 minutes	85	1.6 million

The U-Net architecture took 177 minutes (almost three hours) for complete training, the longest time taken by any network. The U-Net has 31 million learnable parameters which is also the highest number of parameters used by any network in this research. Therefore, besides having only 58 total layers, the learnable parameters are highest among other networks and have the highest crack pixel accuracy after CSN architecture.

The DeepLabv3+ architecture has 100 total layers, the highest number of layers by any network used in this work. The network has 20 million learnable parameters, the second-highest after U-Net. Besides the highest number of layers and second-highest learnable parameters, the DeepLabv3+ has the lowest training time and crack segmentation

accuracy. The network took 33 minutes to complete the training process under set conditions for all the networks. The network did not recognize the pixels efficiently according to their respective classes.

The FCN architecture has 6.9 million learnable parameters, the third-highest learnable parameters by any network used in this work. The network has 43 layers, the lowest number of network layers among all the networks in this work. Besides the lowest number of layers and third-highest learnable parameters, the FCN did not perform well to classify the crack and non-crack pixels accurately. The network took 53 minutes to complete the training process under set conditions for all the networks.

The SegNet architecture has 59 total layers, the third-highest number of layers by any network used in this work. The network has only 0.5 million learnable parameters, the lowest number of learnable parameters by any network in this work. The network took 76 minutes to complete the training process under set conditions for all the networks. The SegNet did not perform efficiently for crack segmentation.

The CSN architecture has 85 total layers, the second-highest number of layers among other networks. The network has 1.6 million learnable parameters, the second-lowest number of learnable parameters. The CSN has a good trade-off between the number of layers and learnable parameters which causes the network to complete the training process in 36 minutes, the second-lowest time taken by any network. The network achieved the highest crack and non-crack pixel accuracy and efficiently tackles the class imbalanced issue for the crack image dataset.

5.5 Statistical Analysis of Test Images

The statistical analysis of all the test images for different loss functions, implemented with FCN, U-Net, SegNet, DeepLabv3+, and CSN are presented through boxplot figures for their mean accuracy. The boxplot explains the distributional characteristics and accuracy of test images. The boxplot has four equal groups of explanation, called quartiles

and each quartile has 25% of all scores. The upper quartile contains 75% of the accuracy whereas the lower half contains 25% of the accuracy. The middle quartile contains 50% of the accuracy of the test images. Figure 5.7 to Figure 5.10 represents the mean accuracy of test images against loss functions through a boxplot. The loss functions are presented on the x-axis and mean accuracy is presented on the y-axis.

Figure 5.7 shows the boxplot that represents the results of mean accuracy against CE, WCE, Dice, Tversky, Focal, and LWF-DTM for FCN architecture. It can be observed that the LWF-DTM in FCN architecture has the highest maximum quartile-3 range and the overall accuracy level is relatively high as compared to other loss functions. The Dice loss function has the longest accuracy bar which shows that the Dice loss function has the lowest accuracy of the test images. The LWF-DTM, WCE, and Tversky loss functions have the highest test image accuracy. The small bar size shows that all the test images have achieved high accuracy.

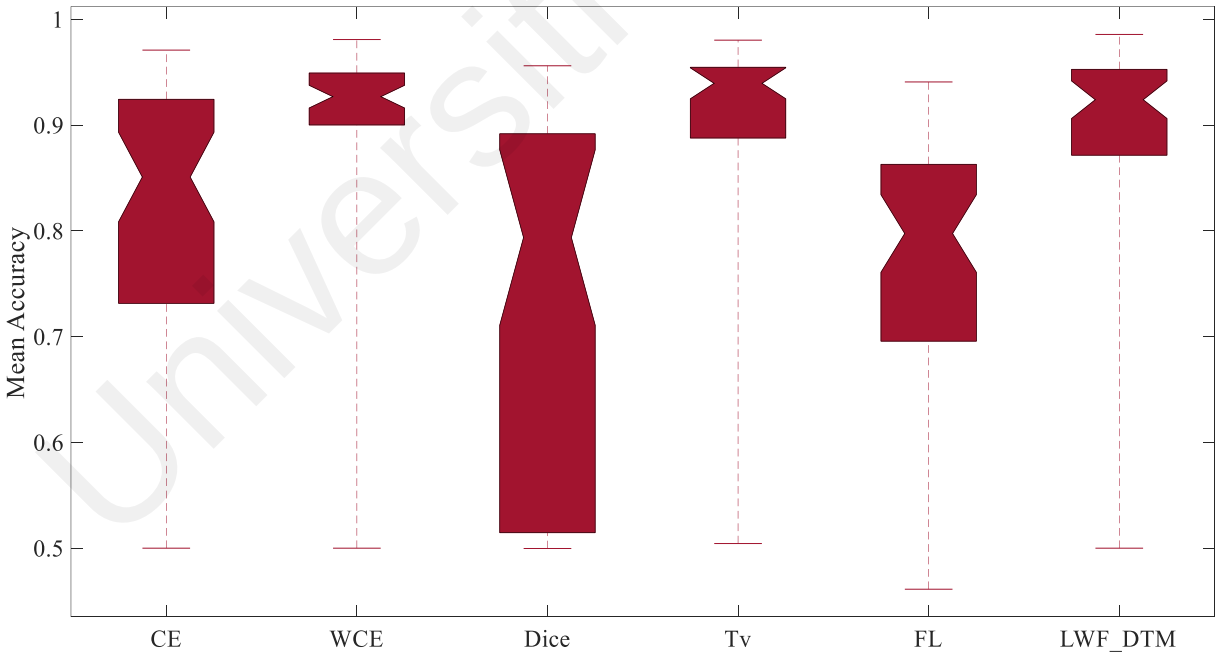

Figure 5.7: BoxPlot of FCN Architecture

Figure 5.8 shows the boxplot that represents the results of mean accuracy against CE, WCE, Dice, Tversky, Focal, and LWF-DTM for U-Net architecture. It can be observed that all the loss functions are performed well, and all the test images have scored more

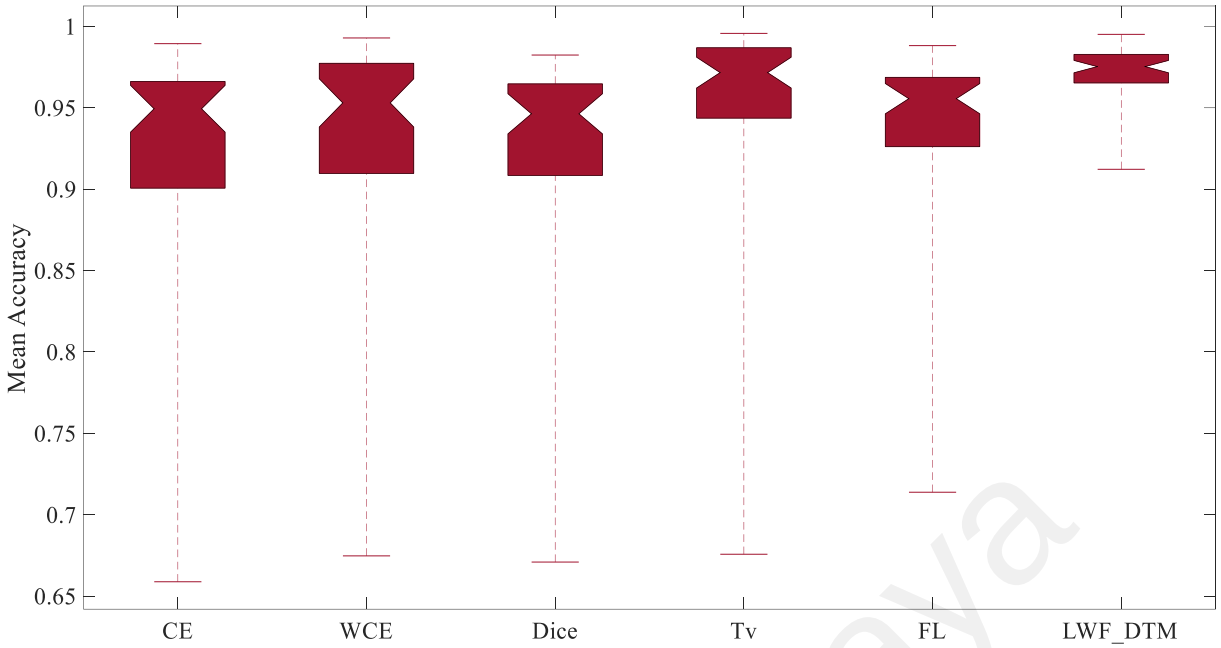

Figure 5.8: BoxPlot of U-Net Architecture

than 90% mean accuracy using U-Net architecture. It can be observed that the LWF-DTM in U-Net architecture has the highest maximum quartile-3 range and the overall accuracy level is relatively high as compared to other loss functions.

Figure 5.9 shows the boxplot that represents the results of mean accuracy against CE, WCE, Dice, Tversky, Focal, and LWF-DTM for SegNet architecture. It can be observed that WCE achieved more than 90% accuracy and LWF-DTM achieved more than 95% accuracy. It can be observed that the LWF-DTM and WCE in SegNet architecture have the highest maximum quartile-3 range and the overall accuracy level is relatively high as

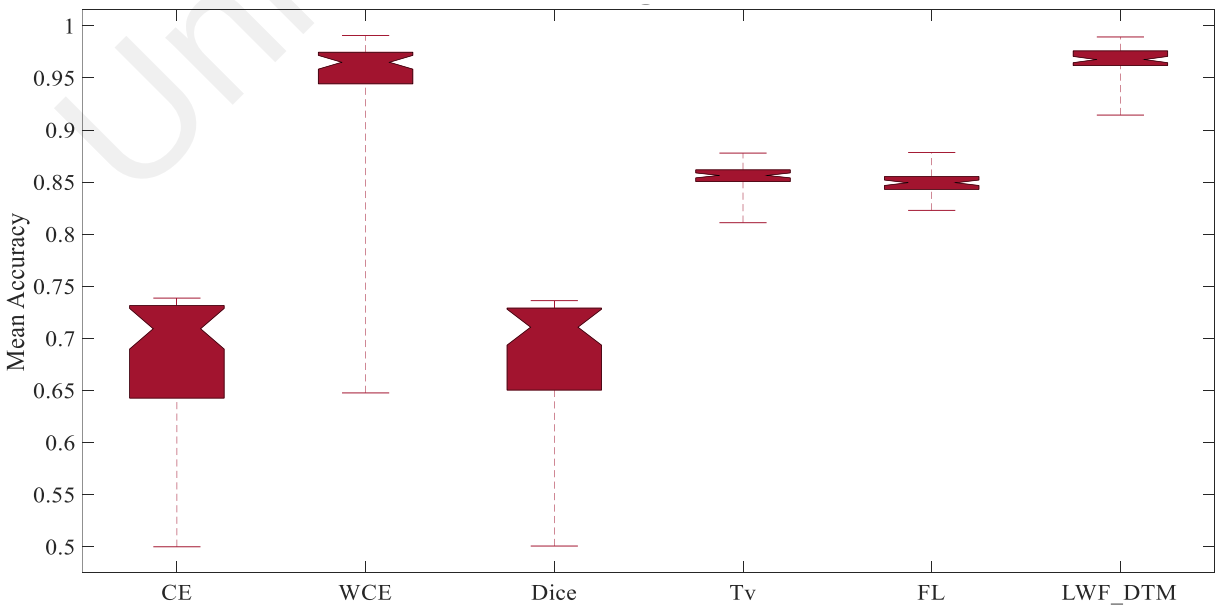

Figure 5.9: BoxPlot of SegNet architecture

compared to other loss functions. The CE and Dice have less than 75% accuracy for all the test images.

Figure 5.11 shows the boxplot that represents the results of mean accuracy against CE, WCE, Dice, Tversky, Focal, and LWF-DTM for DeepLabv3+ architecture. It can be observed that the network has not achieved more than 77% accuracy. WCE and LWF-DTM loss has achieved the highest accuracy for SegNet whereas CE loss scored the lowest accuracy for DeepLabv3.

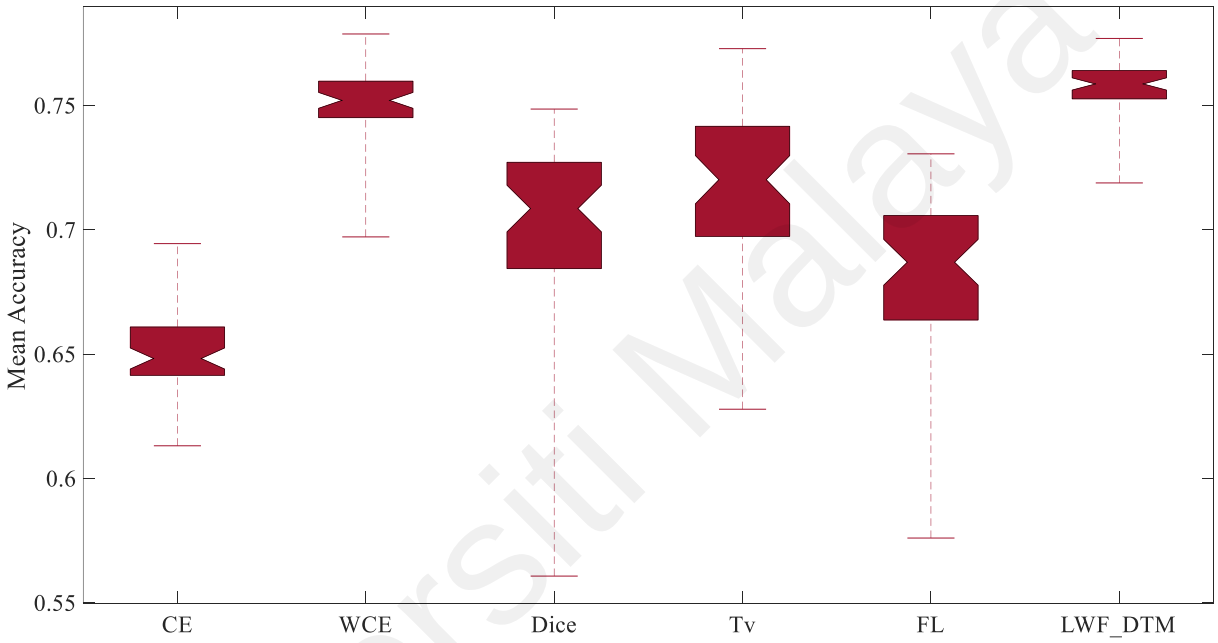

Figure 5.11: BoxPlot of DeepLabv3+ architecture

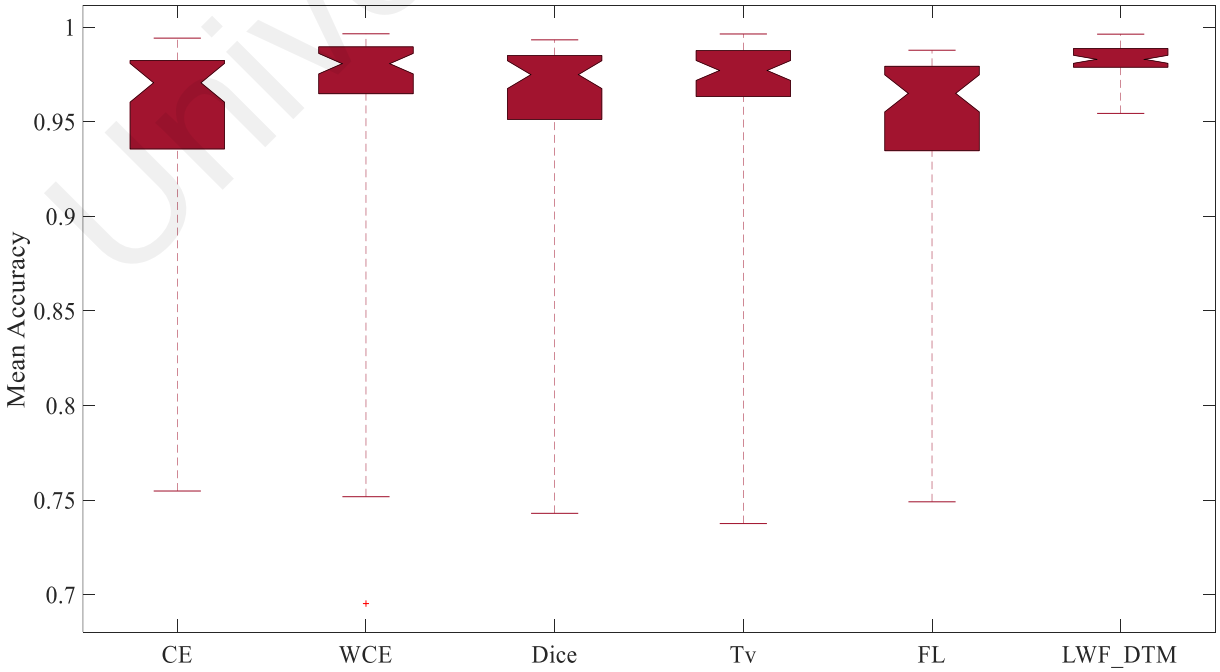

Figure 5.10: BoxPlot of CSN architecture

Figure 5.10 represents the result of mean accuracy against CE, WCE, Dice, Tversky, Focal, and LWF-DTM for CSN architecture. The proposed architecture performed better than other networks implemented in this work. The median value for all the test images lies above 93% of segmentation accuracy. The LWF-DTM in CSN architecture has the highest maximum quartile-3 range, i.e. above 95% and the overall accuracy level is relatively high as compared to other loss functions. LWF-DTM is the only loss function whose all the outliers are inside 95% of mean accuracy while other loss functions have more than one outlier.

5.6 Summary

In this chapter, the results of the crack segmentation network both quantitatively and visually are presented and discussed. The crack segmentation network with LWF-DTM has the potential to segment the crack and non-crack pixels, remove the network biasness due to class imbalance issues, and accurately predict the boundary, spot, stain, and dark intensity pixels. The method is validated using multiple crack segmentation metrics.

CHAPTER 6: CONCLUSION

6.1 Introduction

This chapter summarizes the findings of the crack detection system with respect to the research objectives, which are: (1) investigation of convolutional neural network-based crack detection methods used for crack image segmentation; (2) to build a crack detection system using a CNN that generates a crack segmentation mask; (3) to propose a balancing technique that minimizes the effect of an imbalanced dataset and increases the accuracy of both crack and non-crack pixels; (4) to design and implement crack characteristics measurement technique that measures the crack length, width, area, and orientation; and (5) to evaluate the performance of the proposed crack detection system by comparing the results with state-of-the-art networks and loss functions.

6.2 Conclusions

Structure crack detection is challenging due to several issues such as class imbalance, lighting conditions, complex background, illumination effect, and low contrast between crack and non-crack regions. DL has solved these challenges by automatically extracting the crack and non-crack features from the images and detecting the crack. The advancement in DL has allowed timely detection (classification and segmentation) of cracks on existing civil structures than manual and other detection methods (image processing and ML techniques).

In this research, an extensive review of a potential DL architecture, i.e. CNN on crack detection is provided, and based on this review a crack segmentation network has been devised and implemented for automatic crack detection at the pixel level. The end-to-end encoder-decoder network is built to deal with an imbalanced data set that detects the crack pixels at different orientations and formations. The number of encoder-decoder blocks is set to three. This optimized the computational cost, network complexity, and learning

parameters. An additive attention gate is proposed as a base block between encoder-decoder parts that consists of four different filters which extract precise crack features information whereas the fusion of contextual information from different filters decocts better pixel-level attention. To further minimize the class imbalance issue, a local weighting factor is integrated with the cross-entropy loss function which produces a balanced accuracy for both crack and non-crack pixels. To predict the accurate class pixels from sensitive pixels or confusing regions, a difference transform map is added to penalize or reward the pixels. To examine the performance, a new crack image dataset, namely MSCI, is used for training, validation, and testing. The performance of CSN is tested and compared with FCN, U-Net, SegNet, and DeepLabv3+ network architecture on MSCI crack image datasets. The CSN architecture with LWF-DTM loss function has achieved 98.61% crack pixel accuracy and 98.35% non-crack pixel accuracy on the MSCI dataset. Moreover, the training time has dramatically reduced due to residual blocks in the crack segmentation network.

Furthermore, image processing-based crack characteristics measurement algorithm is proposed to measure the crack orientation, area, length, and width. The algorithm measure 3 values for crack width as crack width varies at each point on the crack, i.e. maximum width, average width, and minimum width. The proposed method significantly classify vertical and horizontal crack but the accuracy of diagonal crack is 85%.

Furthermore, this research also provides the insights obtained through experimental observations of five mostly used semantic segmentation loss functions for crack pixel classification. The performance of five loss functions is evaluated on FCN, U-Net, SegNet, and DeepLabv3+ architecture to find the best performing objective function for crack image segmentation on imbalanced datasets. The LWF-DTM loss function with the local weighting factor has outperformed the other loss functions on CSN architecture. The

imbalanced data effect is well tackled by loss functions with extra weighting factors such as WCE. Other loss functions such as Dice, Tversky, and Focal Tversky mildly handled the class imbalanced effect. Besides the loss functions, the performance of the network architectures is limited by the dataset. Although the CSN architecture shows better performance than other architectures, the values of precision, recall and F1-score still need to be improved.

6.3 Future Works

It is anticipated that DL algorithms will surely replace the conventional methods of detecting cracks. Although the current methods have produced promising results, there is still a need to improve structural crack detection, specifically at pixel-level classification. The research challenges to be considered for future work are described as follows:

1) A standard dataset for the evaluation of network architecture and supporting functions is required. The biasness introduced by the imbalanced dataset affects the network performance, hence a proper weighting method is required to overcome the issue.
2) CNN uses filters on a patch of an image to extract important features and only these features are learned by the model during the training process not the details of each pixel. Therefore, more efficient methods such as transformers can be investigated to perform the crack segmentation task.
3) The network depth is another key factor that can be explored to find the trade-off between the number of layers and learnable parameters. A further study needs to be carried out for better performance with a less computational cost for crack detection.
4) Measurement of crack depth is an important element that determines the crack's severity, and there is no such developed technique that measures the crack depth

using DL. Therefore, a DL-based depth measurement method needs to be explored.

5) DL architectures are heavily parameterized and require large memory and fast processing devices. Real-time implementation of network architecture on hardware is a challenging task. Therefore, hardware implementation is an open research question that can be investigated for real-time crack detection.

REFERENCES

Abdel-Qader, I., Abudayyeh, O., & Kelly, M. E. (2003). Analysis of edge-detection techniques for crack identification in bridges. *Journal of Computing in Civil Engineering*, *17*(4), 255–263. https://doi.org/10.1061/(ASCE)0887-3801(2003)17:4(255)

Abdel-Qader, I., Pashaie-Rad, S., Abudayyeh, O., & Yehia, S. (2006). PCA-based algorithm for unsupervised bridge crack detection. *Advances in Engineering Software*, *37*(12), 771–778. https://doi.org/10.1016/j.advengsoft.2006.06.002

Abraham, N., & Khan, N. M. (2019). A novel focal tversky loss function with improved attention U-Net for lesion segmentation. *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, *2019-April*, 683–687. https://doi.org/10.1109/ISBI.2019.8759329

Adhikari, R. S., Moselhi, O., & Bagchi, A. (2014a). Image-based retrieval of concrete crack properties for bridge inspection. *Automation in Construction*, *39*, 180–194. https://doi.org/10.1016/j.autcon.2013.06.011

Adhikari, R. S., Moselhi, O., & Bagchi, A. (2014b). Image-based retrieval of concrete crack properties for bridge inspection. *Automation in Construction*, *39*, 180–194. https://doi.org/10.1016/j.autcon.2013.06.011

Ahmadi, A., Khalesi, S., & Bagheri, M. (2018a). *Automatic road crack detection and classification using image processing techniques , machine learning and integrated models in urban areas : A novel image binarization technique*. *11*(Iiec), 85–97. http://www.jise.ir/article_69387.html

Ahmadi, A., Khalesi, S., & Bagheri, M. (2018b). Automatic road crack detection and classification using image processing techniques , machine learning and integrated models in urban areas : A novel image binarization technique. *Journal of Industrial and Systems Engineering*, *11*(Iiec), 85–97. http://www.jise.ir/article_69387.html

Ahmed, T. U., Shahadat Hossain, M., Alam, M. J., & Andersson, K. (2019). An integrated CNN-RNN framework to assess road crack. *2019 22nd International Conference on Computer and Information Technology, ICCIT 2019*, *December*, 1–6. https://doi.org/10.1109/ICCIT48885.2019.9038607

Albarqouni, S., Baur, C., Achilles, F., Belagiannis, V., Demirci, S., & Navab, N. (2016). AggNet: Deep learning from crowds for mitosis detection in breast cancer histology images. *IEEE Transactions on Medical Imaging*, *35*(5), 1313–1321. https://doi.org/10.1109/TMI.2016.2528120

Arel, I., Rose, D. C., & Karnowski, T. P. (2010). Deep machine learning a new frontier. *IEEE Computational Intelligence Magazine*, *November*, 13–18. https://doi.org/10.1109/MCI.2010.938364

Arena, A., Delle Piane, C., & Sarout, J. (2014). A new computational approach to cracks quantification from 2D image analysis: Application to micro-cracks description in rocks. *Computers and Geosciences*, *66*, 106–120. https://doi.org/10.1016/j.cageo.2014.01.007

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(12), 2481–2495. https://doi.org/10.1109/TPAMI.2016.2644615

Bang, S., Park, S., Kim, H., & Kim, H. (2019). Encoder–decoder network for pixel-level road crack detection in black-box images. *Computer-Aided Civil and Infrastructure Engineering*, *34*(8), 713–727. https://doi.org/10.1111/mice.12440

Bansal, A., Chen, X., Russell, B., Gupta, A., & Ramanan, D. (2017). *PixelNet: Representation of the pixels, by the pixels, and for the pixels*. http://arxiv.org/abs/1702.06506

Bhatt, P., Sarangi, S., & Pappula, S. (2017). Comparison of CNN models for application in crop health assessment with participatory sensing. *GHTC 2017 - IEEE Global Humanitarian Technology Conference, Proceedings*, *2017-Janua*, 1–7. https://doi.org/10.1109/GHTC.2017.8239295

Bottou, L., & Bousquet, O. (2009). The tradeoffs of large scale learning. *Advances in Neural Information Processing Systems 20 - Proceedings of the 2007 Conference*. https://doi.org/10.7551/mitpress/8996.003.0015

Carr, T. A., Jenkins, M. D., Iglesias, M. I., Buggy, T., & Morison, G. (2018). Road crack detection using a single stage detector based deep neural network. *2018 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS)*, 1–5. https://doi.org/10.1109/EESMS.2018.8405819

Cha, Y., & Choi, W. (2017). Vision-based concrete crack detection using a convolutional neural network. In S. Pakzad (Ed.), *Dynamics of Civil Structures, Volume 2* (Vol. 2, pp. 71–73). Springer International Publishing. https://doi.org/10.1007/978-3-319-54777-0_9

Cha, Y. J., Choi, W., & Büyüköztürk, O. (2017). Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, *32*(5), 361–378. https://doi.org/10.1111/mice.12263

Chaiyasarn, K. (2018). Crack detection in historical structures based on convolutional neural network. *International Journal of GEOMATE*, *15*(51), 240–251. https://doi.org/10.21660/2018.51.35376

Chen, C., Chuah, J. H., Ali, R., & Wang, Y. (2021). Retinal vessel segmentation using deep learning: A review. *IEEE Access*, *9*, 111985–112004. https://doi.org/10.1109/ACCESS.2021.3102176

Chen, F. C., & Jahanshahi, M. R. (2018). NB-CNN: Deep learning-based crack detection using convolutional neural network and naïve bayes data fusion. *IEEE Transactions on Industrial Electronics*, *65*(5), 4392–4400. https://doi.org/10.1109/TIE.2017.2764844

Chen, F. C., & Jahanshahi, M. R. (2020). ARF-Crack: rotation invariant deep fully convolutional network for pixel-level crack detection. *Machine Vision and Applications*, *31*(6). https://doi.org/10.1007/s00138-020-01098-x

Chen, G., Chen, P., Shi, Y., Hsieh, C.-Y., Liao, B., & Zhang, S. (2019). Rethinking the Usage of Batch Normalization and Dropout in the Training of Deep Neural Networks. *ArXiv*. http://arxiv.org/abs/1905.05928

Chen, H., & Murray, A. F. (2003). Continuous restricted Boltzmann machine with an implementable training algorithm. *IEE Proceedings - Vision, Image, and Signal Processing*, *150*(3), 153. https://doi.org/10.1049/ip-vis:20030362

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11211 LNCS*, 833–851. https://doi.org/10.1007/978-3-030-01234-2_49

Chen, T., Cai, Z., Zhao, X., Chen, C., Liang, X., Zou, T., & Wang, P. (2020). Pavement crack detection and recognition using the architecture of segNet. *Journal of Industrial Information Integration*, *18*(March), 100144. https://doi.org/10.1016/j.jii.2020.100144

Cheng, J., Xiong, W., Chen, W., Gu, Y., & Li, Y. (2018). Pixel-level crack detection using U-Net. *TENCON 2018 - 2018 IEEE Region 10 Conference*, *2018-Octob*(October), 0462–0466. https://doi.org/10.1109/TENCON.2018.8650059

Choi, W., & Cha, Y. J. (2020). SDDNet: Real-time crack segmentation. *IEEE Transactions on Industrial Electronics*, *67*(9), 8016–8025. https://doi.org/10.1109/TIE.2019.2945265

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 1–9. http://arxiv.org/abs/1412.3555

Cord, A., & Chambon, S. (2012). Automatic road defect detection by textural pattern recognition based on AdaBoost. *Computer-Aided Civil and Infrastructure Engineering*, *27*(4), 244–259. https://doi.org/10.1111/j.1467-8667.2011.00736.x

Coskun, M., Ucar, A., Yildirim, O., & Demir, Y. (2017). Face recognition based on convolutional neural network. *Proceedings of the International Conference on Modern Electrical and Energy Systems, MEES 2017*, *2018-Janua*, 376–379. https://doi.org/10.1109/MEES.2017.8248937

Costajussà, M. R. (2018). From feature to paradigm: Deep learning in machine translation. *IJCAI International Joint Conference on Artificial Intelligence*, *2018-July*, 5583–5587. https://doi.org/10.1613/jair.1.11198

Courville, Ian Goodfellow, Y. B. and A. (2016). Deep learning. In *Nature* (Vol. 29, Issue 7553). The MIT Press. http://www.deeplearningbook.org

Csurka, G., Larlus, D., & Perronnin, F. (2013). What is a good evaluation measure for semantic segmentation? *Procedings of the British Machine Vision Conference 2013*, 32.1-32.11. https://doi.org/10.5244/C.27.32

Cubero-Fernandez, A., Rodriguez-Lozano, F. J., Villatoro, R., Olivares, J., & Palomares,

J. M. (2017). Efficient pavement crack detection and classification. *EURASIP Journal on Image and Video Processing*, *2017*(1), 39. https://doi.org/10.1186/s13640-017-0187-0

Cui, Y., Jia, M., Lin, T.-Y., Song, Y., & Belongie, S. (2019). Class-Balanced Loss Based on Effective Number of Samples. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, *2019-June*, 9260–9269. https://doi.org/10.1109/CVPR.2019.00949

David Jenkins, M., Carr, T. A., Iglesias, M. I., Buggy, T., & Morison, G. (2018). A deep convolutional neural network for semantic pixel-wise segmentation of road and pavement surface cracks. *2018 26th European Signal Processing Conference (EUSIPCO)*, *2018-Septe*, 2120–2124. https://doi.org/10.23919/EUSIPCO.2018.8553280

Delagnes, P., & Barba, D. (2002). A markov random field for rectilinear structure extraction in pavement distress image analysis. *Proceedings., International Conference on Image Processing*, *1*, 446–449. https://doi.org/10.1109/ICIP.1995.529742

Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., K., A., B., M., Berg, A., & Fei-Fei, L. (2015). Imagenet large scale visual recognition challeng. *International Journal of Computer Vision*, *115*(3), 211–252.

Dias, P. A., Tabb, A., & Medeiros, H. (2018). Apple flower detection using deep convolutional networks. *Computers in Industry*, *99*, 17–28. https://doi.org/10.1016/j.compind.2018.03.010

Dong, Z., Wang, J., Cui, B., Wang, D., & Wang, X. (2020). Patch-based weakly supervised semantic segmentation network for crack detection. *Construction and Building Materials*, *258*, 120291. https://doi.org/10.1016/j.conbuildmat.2020.120291

Dorafshan, S., Thomas, R. J., & Maguire, M. (2018a). Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. *Construction and Building Materials*, *186*, 1031–1045. https://doi.org/10.1016/j.conbuildmat.2018.08.011

Dorafshan, S., Thomas, R. J., & Maguire, M. (2018b). SDNET2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks. *Data in Brief*, *21*, 1664–1668. https://doi.org/10.1016/j.dib.2018.11.015

Dörner, S., Cammerer, S., Hoydis, J., & Ten Brink, S. (2018). Deep learning based communication over the air. *Conference Record of 51st Asilomar Conference on Signals, Systems and Computers, ACSSC 2017*, *2017-Octob*(1), 1791–1795. https://doi.org/10.1109/ACSSC.2017.8335670

Dyrmann, M., Karstoft, H., & Midtiby, H. S. (2016). Plant species classification using deep convolutional neural network. *Biosystems Engineering*, *151*(2005), 72–80. https://doi.org/10.1016/j.biosystemseng.2016.08.024

Eisenbach, M., Stricker, R., Seichter, D., Amende, K., Debes, K., Sesselmann, M.,

Ebersbach, D., Stoeckert, U., & Gross, H.-M. (2017). How to get pavement distress detection ready for deep learning? A systematic approach. *2017 International Joint Conference on Neural Networks (IJCNN)*, *2017-May*, 2039–2047. https://doi.org/10.1109/IJCNN.2017.7966101

Fan, Z., Wu, Y., Lu, J., & Li, W. (2018a). *Automatic pavement crack detection based on structured prediction with the convolutional neural network*. 1–9. http://arxiv.org/abs/1802.02208

Fan, Z., Wu, Y., Lu, J., & Li, W. (2018b). *Automatic pavement crack detection based on structured prediction with the convolutional neural network*. 1–9. http://arxiv.org/abs/1802.02208

Fang, F., Li, L., Gu, Y., Zhu, H., & Lim, J.-H. (2020). A novel hybrid approach for crack detection. *Pattern Recognition*, *107*, 107474. https://doi.org/10.1016/j.patcog.2020.107474

Fei, Y., Wang, K. C. P., Zhang, A., Chen, C., Li, J. Q., Liu, Y., Yang, G., & Li, B. (2019). Pixel-level cracking detection on 3D asphalt pavement images through deep-learning-based crackNet-V. *IEEE Transactions on Intelligent Transportation Systems*, 1–12. https://doi.org/10.1109/TITS.2019.2891167

Feng, C., Zhang, H., Wang, H., Wang, S., & Li, Y. (2020). Automatic pixel-level crack detection on dam surface using deep convolutional network. *Sensors*, *20*(7), 2069. https://doi.org/10.3390/s20072069

Fourcade, A., & Khonsari, R. H. (2019). Deep learning in medical image analysis: A third eye for doctors. *Journal of Stomatology, Oral and Maxillofacial Surgery*, *120*(4), 279–288. https://doi.org/10.1016/j.jormas.2019.06.002

Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, *1*(2), 119–130. https://doi.org/10.1016/0893-6080(88)90014-7

Gehri, N., Mata-Falcón, J., & Kaufmann, W. (2020). Automated crack detection and measurement based on digital image correlation. *Construction and Building Materials*, *256*, 119383. https://doi.org/10.1016/j.conbuildmat.2020.119383

Gers, F., Schraudolph, N. N., & Schmidhuber, J. (2002). Learning Precise Timing with LSTM Recurrent Networks (PDF Download Available). *ResearchGate*, *3*, 115–143. https://www.researchgate.net/publication/220320057_Learning_Precise_Timing_with_LSTM_Recurrent_Networks

Gibb, S., La, H. M., & Louis, S. (2018). A genetic algorithm for convolutional network structure optimization for concrete crack detection. *2018 IEEE Congress on Evolutionary Computation (CEC)*, 1–8. https://doi.org/10.1109/CEC.2018.8477790

Girshick, R. (2015). Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, *2015 Inter*, 1440–1448. https://doi.org/10.1109/ICCV.2015.169

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition, 580–587. https://doi.org/10.1109/CVPR.2014.81

Glorot, X., Bordes, A., & Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. *Proceedings of the 28th International Conference on Machine Learning*, 513–520.

Glorot, Xavier, & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*, *9*, 249–256.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press. http://www.deeplearningbook.org

Gulgec, N. S., Takáč, M., & Pakzad, S. N. (2017). Structural damage detection using convolutional neural networks. In S. Atamturktur, T. Schoenherr, B. Moaveni, & C. Papadimitriou (Eds.), *Model Validation and Uncertainty Quantification, Volume 3* (Vol. 3, Issue June, pp. 331–337). Springer International Publishing. https://doi.org/10.1007/978-3-319-54858-6_33

Gundimeda, V., Kumar, V., C S, A., & Mishra, R. (2019). Traffic light recognition for autonomous vehicles by admixing the traditional ML and DL. In D. P. Nikolaev, P. Radeva, A. Verikas, & J. Zhou (Eds.), *Eleventh International Conference on Machine Vision (ICMV 2018)* (Vol. 11041, p. 88). SPIE. https://doi.org/10.1117/12.2523105

Guo, H., Yu, Y., & Skitmore, M. (2017). Visualization technology-based construction safety management: A review. *Automation in Construction*, *73*, 135–144. https://doi.org/10.1016/j.autcon.2016.10.004

Hadhrami, E. Al, Mufti, M. Al, Taha, B., & Werghi, N. (2018). Ground moving radar targets classification based on spectrogram images using convolutional neural networks. *Proceedings International Radar Symposium*, *2018-June*(Cvd), 1–9. https://doi.org/10.23919/IRS.2018.8447897

Hassan, A., & Mahmood, A. (2017). Deep Learning approach for sentiment analysis of short texts. *2017 3rd International Conference on Control, Automation and Robotics, ICCAR 2017*, *April 2017*, 705–710. https://doi.org/10.1109/ICCAR.2017.7942788

Hayat, M. K., Daud, A., Alshdadi, A. A., Banjar, A., Abbasi, R. A., Bao, Y., & Dawood, H. (2019). Towards deep learning prospects: Insights for social media analytics. *IEEE Access*, *7*(May), 36958–36979. https://doi.org/10.1109/ACCESS.2019.2905101

He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2017). Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*, 2980–2988. https://doi.org/10.1109/ICCV.2017.322

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

He, K., Zhang, X., Ren, S., & Sun, J. (2016b). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Heaton, J. (2018). Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. *Genetic Programming and Evolvable Machines*, *19*(1–2), 305–307.

Hesamian, M. H., Jia, W., He, X., & Kennedy, P. (2019). Deep learning techniques for medical image segmentation: achievements and challenges. *Journal of Digital Imaging*, *32*(4), 582–596. https://doi.org/10.1007/s10278-019-00227-x

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Hu, F., Xia, G. S., Hu, J., & L. Zhang. (2015). Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sens.*, *7*(11), 14680–14707.

Huyan, J., Li, W., Tighe, S., Xu, Z., & Zhai, J. (2020). CrackU-Net: A novel deep convolutional neural network for pixelwise pavement crack detection. *Structural Control and Health Monitoring*, *27*(8), 1–19. https://doi.org/10.1002/stc.2551

Ibragimov, E., Lee, H. J., Lee, J. J., & Kim, N. (2020). Automated pavement distress detection using region based convolutional neural networks. *International Journal of Pavement Engineering*, *0*(0), 1–12. https://doi.org/10.1080/10298436.2020.1833204

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating deep network training by reducing internal covariate shift. *32nd International Conference on Machine Learning, ICML 2015*, *1*, 448–456. http://arxiv.org/abs/1502.03167

Iyer, S., & Sinha, S. K. (2005). A robust approach for automatic detection and segmentation of cracks in underground pipeline images. *Image and Vision Computing*, *23*(10), 921–933. https://doi.org/10.1016/j.imavis.2005.05.017

Jang, K., An, Y.-K., Kim, B., & Cho, S. (2020). Automated crack evaluation of a high-rise bridge pier using a ring-type climbing robot. *Computer-Aided Civil and Infrastructure Engineering*, 1–16. https://doi.org/10.1111/mice.12550

Jang, Y., Son, J., Park, K. H., Park, S. J., & Jung, K.-H. (2018). Laterality Classification of Fundus Images Using Interpretable Deep Neural Network. *Journal of Digital Imaging*, *31*(6), 923–928. https://doi.org/10.1007/s10278-018-0099-2

Ji, J., Wu, L., Chen, Z., Yu, J., Lin, P., & Cheng, S. (2018). Automated pixel-level surface crack detection using U-Net. *Springer Nature Switzerland AG 2018*, *8271*, 69–78. https://doi.org/10.1007/978-3-030-03014-8_6

John S. Miller, & Bellinger, W. Y. (2014). *Distress identification manual for the long-term pavement* (Issue May). www.fhwa.dot.gov/publications/research/infrastructure/pavements/ltpp/13092/13092.pdf

Jung, S. (2019). Semantic vector learning for natural language understanding. *Computer Speech and Language*, *56*, 130–145. https://doi.org/10.1016/j.csl.2018.12.008

Kang, D., Benipal, S. S., Gopal, D. L., & Cha, Y.-J. (2020). Hybrid pixel-level concrete crack segmentation and quantification across complex backgrounds using deep learning. *Automation in Construction*, *118*(June), 103291. https://doi.org/10.1016/j.autcon.2020.103291

Kim, B., & Cho, S. (2018). Automated vision-based detection of cracks on concrete surfaces using a deep learning technique. *Sensors*, *18*(10), 3452. https://doi.org/10.3390/s18103452

Kim, B., & Cho, S. (2020). Automated multiple concrete damage detection using instance segmentation deep learning model. *Applied Sciences (Switzerland)*, *10*(22), 1–17. https://doi.org/10.3390/app10228008

Kim, H., Ahn, E., Shin, M., & Sim, S.-H. (2019). Crack and noncrack classification from concrete surface images using machine learning. *Structural Health Monitoring*, *18*(3), 725–738. https://doi.org/10.1177/1475921718768747

Kim, I.J., & Xie, X. (2015). Handwritten hangul recognition using deep convolutional neural networks. *Nt. J. Doc. Anal. Recognit*, *18*(1), 1–13.

Kim, In Jung, & Xie, X. (2014). Handwritten Hangul recognition using deep convolutional neural networks. *International Journal on Document Analysis and Recognition*, *18*(1), 1–13. https://doi.org/10.1007/s10032-014-0229-4

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15. http://arxiv.org/abs/1412.6980

Koch, C., Paal, S. G., Rashidi, A., Zhu, Z., König, M., & Brilakis, I. (2014). Achievements and challenges in machine vision-based inspection of large concrete structures. *Advances in Structural Engineering*, *17*(3), 303–318. https://doi.org/10.1260/1369-4332.17.3.303

Konig, J., David Jenkins, M., Barrie, P., Mannion, M., & Morison, G. (2019). A convolutional neural network for pavement surface crack segmentation using residual connections and attention gating. *2019 IEEE International Conference on Image Processing (ICIP)*, *2019-Septe*, 1460–1464. https://doi.org/10.1109/ICIP.2019.8803060

Koutsopoulos, H. N., & Downey, A. B. (2006). Primitive-based classification of pavement cracking images. *Journal of Transportation Engineering*, *119*(3), 402–418. https://doi.org/10.1061/(asce)0733-947x(1993)119:3(402)

Krizhevsky, A., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 1–9. https://doi.org/http://dx.doi.org/10.1016/j.protcy.2014.09.007

Kute, R. S., Vyas, V., & Anuse, A. (2019). Component-based face recognition under transfer learning for forensic applications. *Information Sciences*, *476*, 176–191.

https://doi.org/10.1016/j.ins.2018.10.014

Larsson, G., Maire, M., & Shakhnarovich, G. (2016). *FractalNet: Ultra-Deep Neural Networks without Residuals*. 1–11. http://arxiv.org/abs/1605.07648

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

LeCun, Y. (1989). Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems*, 396–404. https://doi.org/10.1111/dsu.12130

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of IEEE*, *86*, 2278–2324. http://ieeexplore.ieee.org/document/726791/#full-text-section

Lee, D., Kim, J., & Lee, D. (2019). Robust concrete crack detection using deep learning-based semantic segmentation. *International Journal of Aeronautical and Space Sciences*, *20*(1), 287–299. https://doi.org/10.1007/s42405-018-0120-5

Lee, H., Pham, P., Largman, Y., & Ng, A. Y. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advance Neural Information Processing System*, *22*, 1096–1104.

Lee, J. S., Hwang, S. H., Choi, I. Y., & Choi, Y. (2020). Estimation of crack width based on shape-sensitive kernels and semantic segmentation. *Structural Control and Health Monitoring*, *27*(4), 1–21. https://doi.org/10.1002/stc.2504

Li, G., Ren, X., Qiao, W., Ma, B., & Li, Y. (2020). Automatic bridge crack identification from concrete surface using ResNeXt with postprocessing. *Structural Control and Health Monitoring*, *27*(11), 1–20. https://doi.org/10.1002/stc.2620

Li, L., Sun, L., Ning, G., & Tan, S. (2014). Automatic pavement crack recognition based on BP neural network. *Promet - Traffic - Traffico*, *26*(1), 11–22. https://doi.org/10.7307/ptt.v26i1.1477

Li, Shengyuan, & Zhao, X. (2019). Image-based concrete crack detection using convolutional neural network and exhaustive search technique. *Advances in Civil Engineering*, *2019*(Ml), 1–12. https://doi.org/10.1155/2019/6520620

Li, Shengyuan, & Zhao, X. (2020). Automatic crack detection and measurement of concrete structure using convolutional encoder-decoder network. *IEEE Access*, *8*, 134602–134618. https://doi.org/10.1109/ACCESS.2020.3011106

Li, Shengyuan, Zhao, X., & Zhou, G. (2019). Automatic pixel-level multiple damage detection of concrete structure using fully convolutional network. *Computer-Aided Civil and Infrastructure Engineering*, *34*(7), 616–634. https://doi.org/10.1111/mice.12433

Li, Shuai, Song, W., Qin, H., & Hao, A. (2018). Deep variance network: An iterative, improved CNN framework for unbalanced training datasets. *Pattern Recognition*,

81, 294–308. https://doi.org/10.1016/j.patcog.2018.03.035

Liang, D., Zhou, X.-F., Wang, S., & Liu, C.-J. (2019). Research on concrete cracks recognition based on dual convolutional neural network. *KSCE Journal of Civil Engineering*, *23*(7), 3066–3074. https://doi.org/10.1007/s12205-019-2030-x

Lin, M., Chen, Q., & Yan, S. (2013). *Network in network*. 1–10. https://doi.org/10.1109/ASRU.2015.7404828

Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollar, P. (2020). Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(2), 318–327. https://doi.org/10.1109/TPAMI.2018.2858826

Lin, W. H., Lin, H. C., Wang, P., Wu, B. H., & Tsai, J. Y. (2018). Using convolutional neural networks to network intrusion detection for cyber threats. *Proceedings of 4th IEEE International Conference on Applied System Innovation 2018, ICASI 2018*, 1107–1110. https://doi.org/10.1109/ICASI.2018.8394474

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

Liu, P. C., & El-Gohary, N. (2019). Automatic annotation of web images for domain-specific crack classification. In I. Mutis & T. Hartmann (Eds.), *Advances in Informatics and Computing in Civil and Construction Engineering* (pp. 553–560). Springer International Publishing. https://doi.org/10.1007/978-3-030-00220-6_66

Liu, Wei, Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9905 LNCS*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

Liu, Wenjun, Huang, Y., Li, Y., & Chen, Q. (2019). *FPCNet: Fast pavement crack detection network based on encoder-decoder architecture*. 1–11. http://arxiv.org/abs/1907.02248

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

Liu, Z., Cao, Y., Wang, Y., & Wang, W. (2019). Computer vision-based concrete crack detection using U-net fully convolutional networks. *Automation in Construction*, *104*(April), 129–139. https://doi.org/10.1016/j.autcon.2019.04.005

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3431–3440. https://doi.org/10.1109/CVPR.2015.7298965

Long, L., Döhler, M., & Thöns, S. (2020). Determination of structural and damage detection system influencing parameters on the value of information. *Structural Health Monitoring*, 147592171990091. https://doi.org/10.1177/1475921719900918

Lottes, P., Behley, J., Milioto, A., & Stachniss, C. (2018). Fully convolutional networks with sequential information for robust crop and weed detection in precision farming. *IEEE Robotics and Automation Letters*, *3*(4), 2870–2877. https://doi.org/10.1109/LRA.2018.2846289

Lu, J., Xu, Y., Chen, M., & Luo, Y. (2018). A coarse-to-fine fully convolutional neural network for fundus vessel segmentation. *Symmetry*, *10*(11), 607. https://doi.org/10.3390/sym10110607

Ma, J. (2020). Segmentation Loss Odyssey. *ArXiv:2005.13449, 2020*. http://arxiv.org/abs/2005.13449

Makinen, Y., Azzari, L., & Foi, A. (2020). Collaborative Filtering of Correlated Noise: Exact Transform-Domain Variance for Improved Shrinkage and Patch Matching. *IEEE Transactions on Image Processing*, *29*, 8339–8354. https://doi.org/10.1109/TIP.2020.3014721

Mandal, V., Uong, L., & Adu-Gyamfi, Y. (2018). Automated road crack detection using deep convolutional neural networks. *2018 IEEE International Conference on Big Data (Big Data)*, 5212–5215. https://doi.org/10.1109/BigData.2018.8622327

Mao, B., Fadlullah, Z. M., Tang, F., Kato, N., Akashi, O., Inoue, T., & Mizutani, K. (2017). Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning. *IEEE Transactions on Computers*, *66*(11), 1946–1960. https://doi.org/10.1109/TC.2017.2709742

MathWorks. (2020). *Computer vision toolbox documentation*. https://www.mathworks.com/help/vision/ref/blobanalysis.html

Milletari, F., Navab, N., & Ahmadi, S. A. (2016). V-Net: Fully convolutional neural networks for volumetric medical image segmentation. *Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016*, 565–571. https://doi.org/10.1109/3DV.2016.79

Mitchell, T. M. (1997). Does machine learning really work? *AI Magazine, 18*(3), 11–20. http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/1303

Mocsari, E., & Stone, S. S. (1978). Densely connected convolutional networks. *American Journal of Veterinary Research*, *39*(9), 1442–1446. https://doi.org/10.1109/CVPR.2017.243

Moussa, G., & Hussain, H. (2011). A new technique for automatic detection and parameters estimation of pavement crack. *Proceedings of the 4th International Multi-Conference on Engineering and Technological Innovation (IMETI 2011)*, *2011*, 1–6. https://doi.org/10.13140/2.1.3191.2001

naceur, M. Ben, Saouli, R., Akil, M., & Kachouri, R. (2018). Fully automatic brain tumor segmentation using end-to-end incremental deep neural networks in MRI images. *Computer Methods and Programs in Biomedicine*, *166*, 39–49. https://doi.org/10.1016/j.cmpb.2018.09.007

Ni, F., Zhang, J., & Chen, Z. (2019). Pixel-level crack delineation in images with

convolutional feature fusion. *Structural Control and Health Monitoring*, *26*(1), e2286. https://doi.org/10.1002/stc.2286

Nugraha, B. T., Su, S. F., & Fahmizal. (2017). Towards self-driving car using convolutional neural network and road lane detector. *Proceedings of the 2nd International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology, ICACOMIT 2017*, *2018-Janua*, 65–69. https://doi.org/10.1109/ICACOMIT.2017.8253388

Nugraha, B. T., Su, S. F., & Fahmizal. (2018). Towards self-driving car using convolutional neural network and road lane detector. *Proceedings of the 2nd International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology, ICACOMIT 2017*, *2018-Janua*, 65–69. https://doi.org/10.1109/ICACOMIT.2017.8253388

Nuruzzaman, M., & Hussain, O. K. (2018). A survey on chatbot implementation in customer service industry through deep neural networks. *Proceedings - 2018 IEEE 15th International Conference on e-Business Engineering, ICEBE 2018*, *August 2019*, 54–61. https://doi.org/10.1109/ICEBE.2018.00019

Oh, J. K., Jang, G., Oh, S., Lee, J. H., Yi, B. J., Moon, Y. S., Lee, J. S., & Choi, Y. (2009). Bridge inspection robot system with machine vision. *Automation in Construction*, *18*(7), 929–941. https://doi.org/10.1016/j.autcon.2009.04.003

Oliveira, H., & Correia, P. L. (2013). Automatic road crack detection and characterization. *IEEE Transactions on Intelligent Transportation Systems*, *14*(1), 155–168. https://doi.org/10.1109/TITS.2012.2208630

Oliveira, H., & Correia, P. L. (2014). CrackIT - An image processing toolbox for crack detection and characterization. *2014 IEEE International Conference on Image Processing, ICIP 2014*, 798–802. https://doi.org/10.1109/ICIP.2014.7025160

Pan, J., Yin, Y., Xiong, J., Luo, W., Gui, G., & Sari, H. (2018). Deep learning-based unmanned surveillance systems for observing water levels. *IEEE Access*, *6*, 73561–73571. https://doi.org/10.1109/ACCESS.2018.2883702

Panella, F., Boehm, J., Loo, Y., Kaushik, A., & Gonzalez, D. (2018). Deep learning and image processing for automated crack detection and defect measurement in underground structures. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII–2*(June 2018), 829–835. https://doi.org/10.5194/isprs-archives-XLII-2-829-2018

Patterson, J., & Gibson, A. (2017). *Deep Learning: A Practitioner's Approach*. "O'Reilly Media, Inc." https://www.safaribooksonline.com/library/view/deep-learning/9781491924570/

Pauly, L., Peel, H., Luo, S., Hogg, D., & Fuentes, R. (2017). Deeper networks for pavement crack detection. *ISARC 2017 - Proceedings of the 34th International Symposium on Automation and Robotics in Construction*, 479–485. https://doi.org/10.22260/ISARC2017/0066

Peng, C., Bu, W., Xiao, J., Wong, K., & Yang, M. (2018). An improved neural network

cascade for face detection in large scene surveillance. *Applied Sciences*, *8*(11), 2222. https://doi.org/10.3390/app8112222

Prasanna, P., Dana, K., Gucunski, N., & Basily, B. (2012). Computer-vision based crack detection and analysis. *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2012*, *8345*, 834542. https://doi.org/10.1117/12.915384

Protopapadakis, E., Voulodimos, A., Doulamis, A., Doulamis, N., & Stathaki, T. (2019). Automatic crack detection for tunnel inspection using deep learning and heuristic image post-processing. *Applied Intelligence*, *49*(7), 2793–2806. https://doi.org/10.1007/s10489-018-01396-y

Rawat, W., & Wang, Z. (2017). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation*, *28*(5), 950–969. https://doi.org/10.1162/NECO

Razavi, S., & Yalcin, H. (2017). Using convolutional neural networks for plant classification. *2017 25th Signal Processing and Communications Applications Conference, SIU 2017*, 1–4. https://doi.org/10.1109/SIU.2017.7960654

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 6517–6525. https://doi.org/10.1109/CVPR.2017.690

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

Ren, Y., Huang, J., Hong, Z., Lu, W., Yin, J., Zou, L., & Shen, X. (2020). Image-based concrete crack detection in tunnels using deep fully convolutional networks. *Construction and Building Materials*, *234*, 117367. https://doi.org/10.1016/j.conbuildmat.2019.117367

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9351*, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2014). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252. http://arxiv.org/abs/1409.0575

Sabour, S., Frosst, N., & Hinton, G. E. (2017). *Dynamic routing between capsules*. *Nips*. http://arxiv.org/abs/1710.09829

Salaken, S. M., Khosravi, A., Nguyen, T., & Nahavandi, S. (2019). Seeded transfer learning for regression problems with deep learning. *Expert Systems with Applications*, *115*, 565–577. https://doi.org/10.1016/j.eswa.2018.08.041

Salehi, S. S. M., Erdogmus, D., & Gholipour, A. (2017). Tversky loss function for image segmentation using 3D fully convolutional deep networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10541 LNCS*, 379–387. https://doi.org/10.1007/978-3-319-67389-9_44

Salman, M., Mathavan, S., Kamal, K., & Rahman, M. (2013). Pavement crack detection using the Gabor filter. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2039–2044. https://doi.org/10.1109/ITSC.2013.6728529

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, *3*(3), 210–229. https://doi.org/10.1147/rd.33.0210

Sapkal, A. M., Munot, M., & Joshi, M. A. (2008). R'G'B' to Y'CbCr color space conversion using FPGA. *IET Conference on Wireless, Mobile and Multimedia Networks*, 255–258. https://doi.org/10.1049/cp:20080191

Shi, Y., Cui, L., Qi, Z., Meng, F., & Chen, Z. (2016a). Automatic road crack detection using random structured forests. *IEEE Transactions on Intelligent Transportation Systems*, *17*(12), 3434–3445. https://doi.org/10.1109/TITS.2016.2552248

Shi, Y., Cui, L., Qi, Z., Meng, F., & Chen, Z. (2016b). Automatic road crack detection using random structured forests. *IEEE Transactions on Intelligent Transportation Systems*, *17*(12), 1–12. https://doi.org/10.1109/TITS.2016.2552248

Shi, Y., Cui, L., Qi, Z., Meng, F., & Chen, Z. (2016c). Automatic road crack detection using random structured forests. *IEEE Transactions on Intelligent Transportation Systems*, *17*(12), 3434–3445. https://doi.org/10.1109/TITS.2016.2552248

Silva, W. R. L. da, & Lucena, D. S. de. (2018). Concrete cracks detection based on deep learning image classification. *Proceedings*, *2*(8), 489. https://doi.org/10.3390/ICEM18-05387

Sinha, S. K., & Fieguth, P. W. (2006). Automated detection of cracks in buried concrete pipe images. *Automation in Construction*, *15*(1), 58–72. https://doi.org/10.1016/j.autcon.2005.02.006

Sobayo, R., Wu, H. H., Ray, R., & Qian, L. (2018). Integration of convolutional neural network and thermal images into soil moisture estimation. *Proceedings - 2018 1st International Conference on Data Intelligence and Security, ICDIS 2018*, 207–210. https://doi.org/10.1109/ICDIS.2018.00041

Socher, R., Lin, C. C.-Y., Manning, C., & Ng., A. Y. (2011). Parsing natural scenes and natural language with recursive neural networks. *Proceedings of the 28th*

International Conference on Machine Learning, 129–136. http://www.socher.org/uploads/Main/SocherLinNgManning_ICML2011.pdf

Song, W., Jia, G., Zhu, H., Jia, D., & Gao, L. (2020). Automated pavement crack damage detection using deep multiscale convolutional features. *Journal of Advanced Transportation*, *2020*(ii), 1–11. https://doi.org/10.1155/2020/6412562

Soon, F. C., Khaw, H. Y., Chuah, J. H., & Kanesan, J. (2019). PCANet-based convolutional neural network architecture for a vehicle model recognition system. *IEEE Transactions on Intelligent Transportation Systems*, *20*(2), 749–759. https://doi.org/10.1109/TITS.2018.2833620

Sreenu, G., & Saleem Durai, M. A. (2019). Intelligent video surveillance: A review through deep learning techniques for crowd analysis. *Journal of Big Data*, *6*(1), 1–27. https://doi.org/10.1186/s40537-019-0212-5

Sudre, C. H., Li, W., Vercauteren, T., Ourselin, S., & Jorge Cardoso, M. (2017). Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 10553 LNCS* (pp. 240–248). https://doi.org/10.1007/978-3-319-67558-9_28

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818–2826. https://doi.org/10.1109/CVPR.2016.308

Taha, A. A., & Hanbury, A. (2015). Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. *BMC Medical Imaging*, *15*(1), 29. https://doi.org/10.1186/s12880-015-0068-x

Taheri, S. (2019). A review on five key sensors for monitoring of concrete structures. *Construction and Building Materials*, *204*, 492–509. https://doi.org/10.1016/j.conbuildmat.2019.01.172

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

Tran, T. S., Tran, V. P., Lee, H. J., Flores, J. M., & Le, V. P. (2020). A two-step sequential automated crack detection and severity classification process for asphalt pavements. *International Journal of Pavement Engineering*, *0*(0), 1–15. https://doi.org/10.1080/10298436.2020.1836561

Tsai, Y.-C., Kaul, V., & Mersereau, R. M. (2009). Critical assessment of pavement distress segmentation methods. *Journal of Transportation Engineering*, *136*(1), 11–

19. https://doi.org/10.1061/(asce)te.1943-5436.0000051

Varadharajan, S., Jose, S., Sharma, K., Wander, L., & Mertz, C. (2014). Vision for road inspection. *2014 IEEE Winter Conference on Applications of Computer Vision, WACV 2014*, 115–122. https://doi.org/10.1109/WACV.2014.6836111

Vinícius dos Santos Ferreira, M., Oseas de Carvalho Filho, A., Dalília de Sousa, A., Corrêa Silva, A., & Gattass, M. (2018). Convolutional neural network and texture descriptor-based automatic detection and diagnosis of glaucoma. *Expert Systems with Applications*, *110*, 250–263. https://doi.org/10.1016/j.eswa.2018.06.010

Vluymans, S. (2019). Learning from imbalanced data. In *Studies in Computational Intelligence* (Vol. 807, Issue 9, pp. 81–110). IEEE. https://doi.org/10.1007/978-3-030-04663-7_4

Vo, T., Nguyen, T., & Le, C. (2018). Race recognition using deep convolutional neural networks. *Symmetry*, *10*(11), 564. https://doi.org/10.3390/sym10110564

Vu, C., & Duc, L. (2019). *Automation in Construction Autonomous concrete crack detection using deep fully convolutional neural network*. *99*(December 2018), 52–58. https://doi.org/10.1016/j.autcon.2018.11.028

Wang, M., & Cheng, J. C. P. (2018). Development and improvement of deep learning based automated defect detection for sewer pipe inspection using faster R-CNN. In I. F. C. Smith & B. Domer (Eds.), *Advanced Computing Strategies for Engineering* (Vol. 10863, pp. 171–192). Springer International Publishing. https://doi.org/10.1007/978-3-319-91638-5_9

Wang, M., & Cheng, J. C. P. (2019). A unified convolutional neural network integrated with conditional random field for pipe defect segmentation. *Computer-Aided Civil and Infrastructure Engineering*, mice.12481. https://doi.org/10.1111/mice.12481

Wang, Xianglong, & Hu, Z. (2017). Grid-based pavement crack analysis using deep learning. *2017 4th International Conference on Transportation Information and Safety (ICTIS)*, 917–924. https://doi.org/10.1109/ICTIS.2017.8047878

Wang, Xinchen, Zhang, W., Wu, X., Xiao, L., Qian, Y., & Fang, Z. (2019). Real-time vehicle type classification with deep convolutional neural networks. In *Journal of Real-Time Image Processing* (Vol. 16, Issue 1, pp. 5–14). https://doi.org/10.1007/s11554-017-0712-5

Wang, Xuejun, & Zhang, Y. (2017). The detection and recognition of bridges' cracks based on deep belief network. *Proceedings - 2017 IEEE International Conference on Computational Science and Engineering and IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, CSE and EUC 2017*, *1*, 768–771. https://doi.org/10.1109/CSE-EUC.2017.151

Xia, B., Cao, J., Zhang, X., & Peng, Y. (2020). Automatic concrete sleeper crack detection using a one-stage detector. *International Journal of Intelligent Robotics and Applications*, *4*(3), 319–327. https://doi.org/10.1007/s41315-020-00141-4

Xu, B., & Huang, Y. (2005). Automatic inspection of pavement cracking distress.

Applications of Digital Image Processing XXVIII, *5909*(1), 590901. https://doi.org/10.1117/12.613770

Xu, H., Su, X., Wang, Y., Cai, H., Cui, K., & Chen, X. (2019). Automatic bridge crack detection using a convolutional neural network. *Applied Sciences*, *9*(14), 2867. https://doi.org/10.3390/app9142867

Xu, L., Lv, S., Deng, Y., & Li, X. (2020). A weakly supervised surface defect detection based on convolutional neural network. *IEEE Access*, *8*, 42285–42296. https://doi.org/10.1109/ACCESS.2020.2977821

Xu, Y., Wang, Y., Yuan, J., Cheng, Q., Wang, X., & Carson, P. L. (2019). Medical breast ultrasound image segmentation by machine learning. *Ultrasonics*, *91*(July 2018), 1–9. https://doi.org/10.1016/j.ultras.2018.07.006

Yamaguchi, T., Nakamura, S., Saegusa, R., & Hashimoto, S. (2008). Image-based crack detection for real concrete surfaces. *IEEJ Transactions on Electrical and Electronic Engineering*, *3*(1), 128–135. https://doi.org/10.1002/tee.20244

Yamane, T., & Chun, P. (2020). Crack detection from a concrete surface image based on semantic segmentation using deep learning. *Journal of Advanced Concrete Technology*, *18*(9), 493–504. https://doi.org/10.3151/jact.18.493

Yang, L., Li, B., Li, W., Liu, Z., Yang, G., & Xiao, J. (2017). A robotic system towards concrete structure spalling and crack database. *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, *2018-Janua*, 1276–1281. https://doi.org/10.1109/ROBIO.2017.8324593

Yang, X., Li, H., Yu, Y., Luo, X., Huang, T., & Yang, X. (2018). Automatic pixel-level crack detection and measurement using fully convolutional network. *Computer-Aided Civil and Infrastructure Engineering*, *33*(12), 1090–1109. https://doi.org/10.1111/mice.12412

Yokoyama, S., & Matsumoto, T. (2017). Development of an automatic detector of cracks in concrete using machine learning. *Procedia Engineering*, *171*, 1250–1255. https://doi.org/10.1016/j.proeng.2017.01.418

Zeiler, M. D., & Fergus, R. (2013). *Visualizing and understanding convolutional networks*. https://doi.org/10.1007/978-3-319-10590-1_53

Zeng, H., Yang, C., Dai, G., Qin, F., Zhang, J., & Kong, W. (2018). EEG classification of driver mental states by deep learning. *Cognitive Neurodynamics*, *12*. https://doi.org/10.1007/s11571-018-9496-y

Zeng, M.-H., Wu, Z.-M., & Wang, Y.-J. (2020). A stochastic model considering heterogeneity and crack propagation in concrete. *Construction and Building Materials*, *254*, 119289. https://doi.org/10.1016/j.conbuildmat.2020.119289

Zhang, A., Wang, K. C. P., Fei, Y., Liu, Y., Chen, C., Yang, G., Li, J. Q., Yang, E., & Qiu, S. (2019). Automated pixel-level pavement crack detection on 3D asphalt surfaces with a recurrent neural network. *Computer-Aided Civil and Infrastructure Engineering*, *34*(3), 213–229. https://doi.org/10.1111/mice.12409

Zhang, A., Wang, K. C. P., Fei, Y., Liu, Y., Tao, S., Chen, C., Li, J. Q., & Li, B. (2018). Deep learning–based fully automated pavement crack detection on 3D asphalt surfaces with an improved CrackNet. *Journal of Computing in Civil Engineering*, *32*(5), 04018041. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000775

Zhang, A., Wang, K. C. P., Li, B., Yang, E., Dai, X., Peng, Y., Fei, Y., Liu, Y., Li, J. Q., & Chen, C. (2017). Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network. *Computer-Aided Civil and Infrastructure Engineering*, *32*(10), 805–819. https://doi.org/10.1111/mice.12297

Zhang, Kai, Zuo, W., & Zhang, L. (2018). FFDNet: Toward a Fast and Flexible Solution for CNN-Based Image Denoising. *IEEE Transactions on Image Processing*, *27*(9), 4608–4622. https://doi.org/10.1109/TIP.2018.2839891

Zhang, Kaige, Cheng, H.-D., & Gai, S. (2018). Efficient dense-dilation network for pavement cracks detection with large input image size. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, *2018-Novem*, 884–889. https://doi.org/10.1109/ITSC.2018.8569958

Zhang, Lei, Yang, F., Daniel Zhang, Y., & Zhu, Y. J. (2016a). Road crack detection using deep convolutional neural network. *Proceedings - International Conference on Image Processing, ICIP*, *2016-Augus*, 3708–3712. https://doi.org/10.1109/ICIP.2016.7533052

Zhang, Lei, Yang, F., Daniel Zhang, Y., & Zhu, Y. J. (2016b). Road crack detection using deep convolutional neural network. *2016 IEEE International Conference on Image Processing (ICIP)*, *2016-Augus*, 3708–3712. https://doi.org/10.1109/ICIP.2016.7533052

Zhang, Lingxin, Shen, J., & Zhu, B. (2020). A research on an improved UNet-based concrete crack detection algorithm. *Structural Health Monitoring*, *29*, 147592172094006. https://doi.org/10.1177/1475921720940068

Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning based recommender system. *ACM Computing Surveys*, *52*(1), 1–38. https://doi.org/10.1145/3285029

Zhang, X., Rajan, D., & Story, B. (2019). Concrete crack detection using context-aware deep semantic segmentation network. *Computer-Aided Civil and Infrastructure Engineering*, *34*(11), 951–971. https://doi.org/10.1111/mice.12477

Zhang, Y., Chen, B., Wang, J., Li, J., & Sun, X. (2020). APLCNet: Automatic pixel-level crack detection network based on instance segmentation. *IEEE Access*, *8*, 199159–199170. https://doi.org/10.1109/access.2020.3033661

Zhao, X., & Li, S. (2018). Convolutional neural networks-based crack detection for real concrete surface. In H. Sohn (Ed.), *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2018* (Issue March, p. 143). SPIE. https://doi.org/10.1117/12.2296536

Zou, Q., Cao, Y., Li, Q., Mao, Q., & Wang, S. (2012). CrackTree: Automatic crack detection from pavement images. *Pattern Recognition Letters*, *33*(3), 227–238. https://doi.org/10.1016/j.patrec.2011.11.004

Zou, Q., Zhang, Z., Li, Q., Qi, X., Wang, Q., & Wang, S. (2018). DeepCrack: Learning hierarchical convolutional features for crack detection. *IEEE Transactions on Image Processing*, *28*(3), 1498–1512. https://doi.org/10.1109/TIP.2018.2878966