UNIVERSITY OF SALERNO

Department of Civil Engineering

PhD Course
on
Risk and Sustainability
in Civil, Architectural and Environmental Engineering Systems

XXXV Cycle (2019-2022)

Machine Learning for Crack Segmentation from Photogrammetric Imagery

Lucas Matias Gujski

[illegible]

Tutor	Co-Tutor	Coordinator
Prof. Margherita Fiani	*Prof. Salvatore Barba*	*Prof. Fernando Fraternali*
[illegible]	[illegible]	[illegible]

[Page reserved for the Dedication]

Summary

List of Figures

List of Tables

List of abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DL	Deep Learning
F1	F1 Score, an evaluation metric
GNSS	Global navigation satellite system
GPU	Graphics Processing Unit
GSD	Ground Sampling Distance
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
ML	Machine Learning
MLS	Mobile Laser Scanning
MVS	Multi-View Stereo
RAM	Random Access Memory
ReLU	Rectified Linear Unit
RTK	Real-time kinematic positioning
SfM	Structure from Motion
TLS	Terrestrial Laser Scanning
UAV	Unmanned Airborne Vehicle

Abstract

The aim of the study proposed in this thesis was to analyse and implement data processing procedures and algorithms to try to overcome the criticalities present in the traditional identification and segmentation of cracks on road pavements and buildings.

For this purpose, algorithms were implemented in Python in order to optimise, on the one hand, the point cloud and products from the photogrammetric process and, on the other hand, the crack segmentation methodology, which is currently the most accurate in the literature.

Point clouds produced by photogrammetric software are not directly usable, as they must first be processed to remove outliers and noise. The first phase of the thesis presents an innovative approach that can assist survey methods by applying an AI algorithm to improve the accuracy of point clouds generated from UAV images.

Many studies on the semantic segmentation of cracks using Machine Learning and Deep Learning techniques can be found in the relevant literature. However, this task is very challenging due to the complexity of the background, as cracks are easily confused with objects not belonging to the surface, shadows, and background textures and are also inhomogeneous.

The results obtained to date are quite good, but often the accuracy of the trained model and the results achieved are evaluated using traditional metrics only. In most cases, the goal is merely to detect the occurrence of cracks. Particular attention should be paid to the thickness of the segmented crack, as the width of the crack is the main parameter for maintenance and characterizes the severity levels. The aim of our study is to optimize the crack segmentation process through the implementation of a modified U-Net model-based architecture. U-Net is a network with two symmetrical branches (encoder-decoder structure). The encoder is replaced with a ResNet50 encoder pre-trained on the ImageNet dataset. Our focus was on crack segmentation, and for this purpose, we used the Crack500 Dataset and compared the results with those obtained from the algorithm currently considered the most accurate and performant in the literature.

To demonstrate the generalization of the model, two real case studies were tested by performing a UAV survey to obtain the photogrammetric models of both.

The results are promising and accurate, with the shape and width of the segmented cracks closely resembling reality.

Riassunto

Lo scopo dello studio proposto in questa tesi è stato quello di analizzare e implementare particolari procedure e algoritmi di elaborazione dei dati per superare le criticità nella tradizionale identificazione e segmentazione delle fessure nelle pavimentazioni stradali ed edilizie.

A tal fine, sono stati implementati algoritmi Python per ottimizzare la nuvola di punti e i prodotti del processo fotogrammetrico, nonché la metodologia di segmentazione delle fessure, la più accurata in letteratura.

Le nuvole di punti prodotte dai software fotogrammetrici non sono direttamente utilizzabili, ma devono essere elaborate per rimuovere gli outlier e il rumore. La prima fase della tesi presenta un approccio innovativo per aiutare i metodi di rilievo utilizzando un algoritmo di intelligenza artificiale che migliora l'accuratezza delle nuvole di punti generate da immagini UAV.

In letteratura si trovano studi sulla segmentazione semantica delle fessure utilizzando tecniche di Machine Learning e Deep Learning. Tuttavia, questo compito è difficile a causa della complessità dello sfondo, dove le fessure possono essere confuse con oggetti non appartenenti alla superficie, ombre e texture dello sfondo, oltre a non essere omogenee.

I risultati ottenuti finora sono ottimi, ma l'accuratezza del modello addestrato e i risultati sono valutati utilizzando metriche tradizionali. L'obiettivo è rilevare la presenza di fessure e fare attenzione allo spessore della fessura segmentata, poiché la sua larghezza è un parametro chiave per la manutenzione e i livelli di gravità. L'obiettivo del nostro studio è quello di ottimizzare il processo di segmentazione delle fessure attraverso l'implementazione di un'architettura basata sul modello U-Net modificato. U-Net è una rete con due rami simmetrici (struttura encoder-decoder). L'encoder è stato sostituito con un encoder ResNet50 pre-addestrato sul set di dati ImageNet. La nostra attenzione si è concentrata sulla segmentazione delle fessure e a tal fine abbiamo utilizzato il dataset Crack500, confrontando i risultati con quelli ottenuti dall'algoritmo attualmente considerato il più accurato e performante in letteratura.

Per dimostrare la generalizzazione del modello, sono stati testati due casi di studio reali eseguendo un rilievo UAV per ottenere i modelli fotogrammetrici di entrambi.

I risultati sono promettenti e accurati, con forma e larghezza delle fessure segmentate molto simili alla realtà.

Resumen

El objetivo del estudio propuesto en esta tesis fue analizar e implementar procedimientos y algoritmos particulares de procesamiento de datos con el fin de intentar superar las criticidades presentes en la tradicional identificación y segmentación de grietas en pavimentos de carreteras y edificios.

Para ello, se implementaron algoritmos en Python para optimizar la nube de puntos y los productos del proceso fotogramétrico, así como la metodología de segmentación de grietas, la más precisa de la literatura.

Las nubes de puntos producidas por el software fotogramétrico no son utilizables directamente, deben procesarse para eliminar valores atípicos y ruido. La primera fase de la tesis presenta un enfoque innovador para ayudar a los métodos de levantamiento mediante un algoritmo de IA que mejora la precisión de las nubes de puntos generadas desde imágenes de UAV.

En la literatura se encuentran estudios sobre la segmentación semántica de grietas usando técnicas de Machine Learning y Deep Learning. Sin embargo, esta tarea es difícil debido a la complejidad del fondo, donde las grietas pueden confundirse con objetos ajenos a la superficie, sombras y texturas de fondo, y además no son homogéneas.

Los resultados conseguidos hasta la fecha son óptimos, pero a menudo la precisión del modelo entrenado y los resultados obtenidos se evalúan utilizando únicamente métricas tradicionales. En la mayoría de los casos, el objetivo es simplemente detectar la aparición de grietas. Debe prestarse especial atención al grosor de la grieta segmentada, ya que la anchura de la misma es el principal parámetro para el mantenimiento y caracteriza los niveles de gravedad. El objetivo de nuestro estudio es optimizar el proceso de segmentación de grietas mediante la aplicación de una arquitectura modificada basada en el modelo U-Net. U-Net es una red con dos ramas simétricas (estructura codificador-decodificador). El codificador se sustituye por un codificador ResNet50 preentrenado en el conjunto de datos ImageNet. Nos centramos en la segmentación de grietas y, para ello, utilizamos el conjunto de datos Crack500 y comparamos los resultados con los obtenidos con el algoritmo considerado actualmente el más preciso y eficaz de la bibliografía.

Para demostrar la generalización del modelo, se probaron dos casos de estudio reales realizando un levantamiento con UAV para obtener modelos fotogramétricos.

Los resultados son prometedores y precisos, mientras que la forma y la anchura de las grietas segmentadas se asemejan mucho a la realidad.

1. Introduction

1.1 Motivation

The detection of cracks in structures and infrastructures is a critical task in civil engineering, as it plays a crucial role in ensuring the integrity and safety of various structures, such as buildings, bridges, tunnels, and roads, that may need to be closed off to public access.

The assessment and examination involved in structural health monitoring (SHM) typically complement maintenance procedures. This is a critical step that takes into account the actual condition of structures and identifies any existing anomalies. Traditionally, visual inspection and manual interpretation have been employed for crack detection. However, these methods are time-consuming, subjective, and susceptible to errors. Moreover, this task can be laborious as it is often performed on-site, sometimes in hazardous situations, and involves difficult-to-access structures. This may result in infrastructure downtime. The identification of cracks can be carried out on 3D or 2D data and is aimed at the assessment of their severity levels. Based on the severity levels, the management entity will be able to draw up an efficient maintenance plan [1].

The acquisition of 3D data has traditionally been a challenging and resource-intensive process, relying on expensive and complex LiDAR technology. Processing these data sets can be time-consuming and costly, especially when working with a large volume of data points [2]. In comparison, 2D data acquisition, such as images, leverages more affordable systems, leading to significantly reduced processing times [3]. This cost and time efficiency has encouraged researchers to investigate high-performance algorithms to identify and segment cracks from images, with a focus on improving accuracy and reducing computational requirements [4].

In the last decade, the application of photogrammetry in digital 3D recording has experienced significant growth. This surge can be attributed to advancements in computer vision technology and the emergence of new computing capabilities that have revolutionized the field. As a result, photogrammetry technology has seen substantial improvements in processing speed and the introduction of automation, effectively addressing a long-standing weakness that once plagued the field [5]. These enhancements have made photogrammetry a more viable and efficient alternative for 3D data acquisition and analysis, expanding its potential applications across various industries.

With the use of automatic Structure from Motion technology (SfM), the overall situation has gradually changed from the widespread use of scientific applications of 3D measurement using laser scanner technology to the use of photogrammetry. Nowadays, photogrammetry technology has gained greater "robustness", possibly overcoming distance-based sensors in many applications. Recently, the technological development of Unmanned Aerial Vehicles (UAV) has become easier to pilot and more reliable, which indirectly promotes the growth of photogrammetric applications, especially in medium and large applications.

In recent years, machine learning algorithms have been increasingly utilized for crack segmentation in various materials. This approach has been shown to have several advantages over traditional methods. Firstly, machine learning algorithms can handle large amounts of data quickly and accurately without the need for human intervention. This is especially useful when dealing with complex structures or large datasets, as it reduces the time and effort required for manual analysis. Additionally, machine learning models can learn from past experiences and improve their performance over time, which can further enhance their accuracy and efficiency.

Moreover, machine learning algorithms can adapt to different types of crack patterns and surface textures, making them more versatile than traditional methods. They can also be trained on specific datasets, allowing them to accurately identify different types of cracks in various materials, including concrete, steel, and asphalt.

The use of machine learning algorithms can significantly reduce costs, as it eliminates the need for specialized equipment or manual labour. This makes crack segmentation more accessible to researchers, engineers, and maintenance professionals, allowing them to perform more frequent and accurate inspections on structures.

Overall, the integration of machine learning algorithms into crack segmentation is a promising approach that has the potential to revolutionize the field of structural health monitoring, enabling more efficient, accurate, and cost-effective evaluation of structures' integrity.

An advancement in the identification of cracks was made thanks to deep learning techniques. Deep learning (DL), a branch of artificial intelligence (AI), has been very successful in semantic segmentation. The goals of crack segmentation can be met greatly by the semantic segmentation technique, which predicts a classification label for each pixel of the image.

The proposed approach involves capturing high-resolution images of the surface of a structure, such as pavement or a building. These images, utilizing photogrammetry, are then used to generate orthophotos and 3D models. The images are also fed into a convolutional neural network (CNN). The CNNs, a key subfield in DL, provide promising results in the pixel-level detection of target objects in noisy images [6,7]. Another benefit of the CNN's end-to-end segmentation method is that it significantly reduces the need for human involvement in the segmentation process. Compared to the manually created features utilized in conventional approaches, the features obtained by convolutional neural networks offer better rendering performance. By using this deep learning capability, certain efforts are devoted to developing reliable feature representations for segmenting images of cracks. Deep neural networks use characteristics to identify whether or not there are cracks in the patches of a picture by combining the pixel-level classification confidence from several frames with various lighting conditions.

This thesis suggests a technique for semantic crack image segmentation based on the residual structure developed on the architecture of the U-Net model with a Res-Net50 encoder. Without first identifying the region of interest, this approach can automatically segment the cracks in images with a complex background, independently learning the characteristics of the cracks and obtaining additional feature information.

While identifying cracks in images is a crucial first step, this alone does not provide sufficient information about their severity. From the analysis, it can be inferred that a wide range of scientific articles focuses on crack detection only. However, calculating the dimensions of the crack was not performed in the majority of the studies. By using photogrammetry to create a three-dimensional model of the crack, we can accurately estimate its shape and size. This information can then be used to determine the appropriate maintenance required to address the crack.

1.2 Objectives

Deep learning techniques have proven to be effective in various applications, including image crack segmentation. The workflow and objectives of this thesis can be summarised in several steps, starting with the selection of a suitable deep learning model, such as a UNet-based architecture. This model is specifically designed for crack segmentation using images. To improve the model's generalization and increase the size of the training dataset, data augmentation techniques can be employed.

In parallel, we can conduct a survey data collection process to identify representative surfaces that exhibit various types of cracks. Cameras mounted on drones can be used to capture high-resolution images of the surfaces. These images will then serve as input for photogrammetric analysis using Structure from Motion techniques. The SfM approach allows for the reconstruction of the 3D geometry of the surfaces from the collected images.

Once the 3D geometry is obtained, the next step involves optimizing the generated point clouds. This can be achieved by calculating photogrammetric accuracy parameters, such as the reprojection error, the angle of intersection between homologous rays, the number of cameras used for each single tie point calculation, and the projection accuracy. Noise and outliers points can be removed from the point clouds using the accuracy parameters and K-Means, wich is a machine learning algorithm.

After optimizing the point clouds, the trained deep learning model can be applied to segment cracks in real case studies. To assess the accuracy of the segmented cracks, the measurement between the predicted and actual cracks must be properly verified. This process involves performing image-to-point cloud association, which matches corresponding features in a 2D image with points in a 3D point cloud. Relevant features and metrics can then be extracted from the detected cracks for further analysis and interpretation.

By following these steps, researchers and engineers can develop a comprehensive understanding of various types of cracks, ultimately informing more effective crack detection and prevention strategies.

1.3 Outline

The structure of this thesis is divided as follows. Chapter 1 is an introduction to the thesis, with motivation and proposed objectives to be achieved.

Chapter 2 gives an overview of machine learning and deep learning and clarifies the terms used throughout this thesis.

In Chapter 3, we review the literature concerning crack monitoring, with a focus on semantic segmentation using images.

Chapter 4 presents the workflow of the proposed U-net-based model, employing ResNet50 as the encoder. Section 4.1 describes the dataset used, Section 4.2 outlines the network architecture, Section 4.3 explains the metrics used to evaluate the model, and Section 4.4 presents the results of the model on the dataset.

Chapter 5 is divided into two major sections. Section 5.1 begins with an overview of the methods used and how they relate to the proposed CNN model (Chapter 4). Subsection 5.1.1 describes the use of K-Means as a machine learning algorithm for point cloud filtering and the accuracy parameters employed. Subsection 5.1.2 details the method for crack measurement, and Subsection 5.1.3 outlines the process of associating crack images with the point cloud produced by photogrammetric software. Section 5.2, the second major part of Chapter 5, is further divided into several subsections. Subsection 5.2.1 introduces the case studies and presents the equipment used for photogrammetric surveying. Subsection 5.2.2 demonstrates the application of the proposed model to the images from these case studies. The use of K-Means for cloud filtering in the case studies is discussed in Subsection 5.2.3. Subsection 5.2.4 validates the metrics of the cracks using measurements taken on-site with a calliper. Subsection 5.2.5 presents the results of associating the crack images with the point cloud as applied to the two case studies.

Finally, the concluding chapter summarizes the observations and mentions the contributions made by the research and suggests a perspective of possible developments that could be addressed in the future.

2. Background: Machine Learning & Deep Learning

Artificial intelligence (AI) has advanced rapidly in recent years, with machine learning (ML) and deep learning (DL) playing a critical role in driving these developments. Artificial intelligence, machine learning, and deep learning are interconnected fields that have demonstrated remarkable progress and have transformed various domains, such as computer vision, natural language processing, and speech recognition [8].

AI is a branch of computer science that aims to create systems capable of performing tasks that typically require human intelligence, such as reasoning, learning, problem-solving, and perception [9]. AI research encompasses various approaches, including knowledge representation, planning, robotics, natural language processing, and ML.

Machine learning (ML), depicted in Figure 2.1, is a subset of artificial intelligence (AI) that enables computers to learn from data and make predictions or decisions without specific programming. Instead, it relies on patterns and inference, which form the very basis of ML [10]. Algorithms build a model based on sample data, known as "training data", to make predictions and perform classifications.

The presence of more data eases the process of recognizing patterns and inferences. Today, we are witnessing a surge in data volume and variety, accompanied by more affordable and powerful computational processes. This environment facilitates the creation of models capable of quickly processing large-scale data, thus providing more accurate results. This, in turn, is generating a growing interest in machine learning. The learning process in ML can be categorized into supervised, unsupervised, and reinforcement learning, each having its own set of techniques and applications [11].

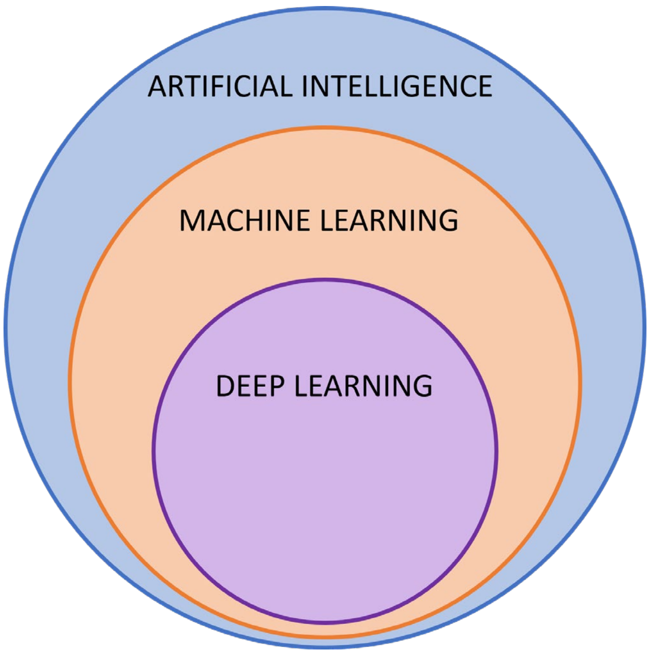


Figure 2.1 Diagram - Subsets of AI

Supervised learning is a common ML approach that involves learning a mapping between input features and target outputs based on labelled data. Training data, comprising a set of training examples, is fundamental to supervised learning. Each training example includes one or more inputs along with the desired output, also known as a supervisory signal [9]. Supervised learning is frequently used in applications where historical data is utilized to predict likely future events. For instance, it can be employed to anticipate fraudulent credit card transactions or to predict which insurance customer is likely to file a claim. Popular algorithms include linear regression, support vector machines (SVM) [12], decision trees [13], and methods such as random forests [14] and gradient boosting machines (GBM) [15].

Unsupervised learning aims to discover underlying patterns or structures in unlabeled data. In the learning process, the algorithm scans through datasets to find meaningful connections. However, acquiring labelled data can be both time-consuming and expensive, which is where unsupervised machine learning comes into play. Unsupervised learning is particularly effective with transactional data. For instance, it can identify segments of customers with similar attributes who can then be targeted collectively in marketing campaigns. Alternatively, it can discern the main attributes that distinguish different customer segments from each other. Widely-used techniques include clustering algorithms, such as K-Means [16,17] and hierarchical clustering [18], as well as dimensionality reduction methods like principal component analysis (PCA) [19] and t-distributed stochastic neighbour embedding (t-SNE) [20].

Reinforcement learning (RL) focuses on training agents to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. Unlike supervised learning, reinforcement learning doesn't require labelled input/output pairs, nor does it need explicit correction of sub-optimal actions. Instead, it focuses on striking a balance between exploration, which involves venturing into uncharted territory, and exploitation, which leverages current knowledge [21]. Q-learning [22] and policy gradients [23] are two popular RL algorithms.

Deep learning, a subset of ML (Figure 2.1), has emerged as a powerful tool for various tasks in computer vision, such as image classification, object detection, and image segmentation [8]. DL is distinguished by its utilization of multi-layered artificial neural networks (ANNs), which facilitate the extraction of intricate, hierarchical attributes from raw data.

Artificial neural networks (ANNs) mimic the human brain's problem-solving abilities by comprising a network of interconnected neurons. These networks draw inspiration from biological neurons, which process input signals and activate when the aggregated input surpasses a specific threshold, subsequently sending an output signal from the axon to other neurons. Similarly, an artificial neuron accepts inputs from preceding neurons, integrates them, and routes the composite signal via an activation function, allowing the output to advance to the next neuron.

It is worth noting that artificial neurons are not all directly interconnected. Instead, they are arranged into groups called layers, which typically establish sequential connections with subsequent layers (Figure 2.2). The input signal navigates through the network from the input layer to the output layer, crossing intermediate layers known as hidden layers. These hidden layers (Figure 2.2) contribute to the ANN's ability to learn complex patterns and representations, enabling the network to tackle intricate problem-solving tasks and adapt to various applications, such as image recognition, natural language processing, and reinforcement learning. Some popular deep learning architectures include recurrent neural networks (RNNs) and convolutional neural networks (CNNs).

RNNs, proposed by Rumelhart et al. in 1986 [24], are a class of ANNs specifically designed to model sequences and temporal dependencies. In contrast to feedforward networks, RNNs possess connections between hidden units that form directed cycles, allowing them to maintain a hidden state capable of storing information from previous time steps.

This property renders RNNs well-suited for tasks involving sequential data, such as language modelling, speech recognition, and time-series prediction [25,26]. However, RNNs are prone to vanishing and exploding gradient issues, making it challenging to learn long-range dependencies [27,28].

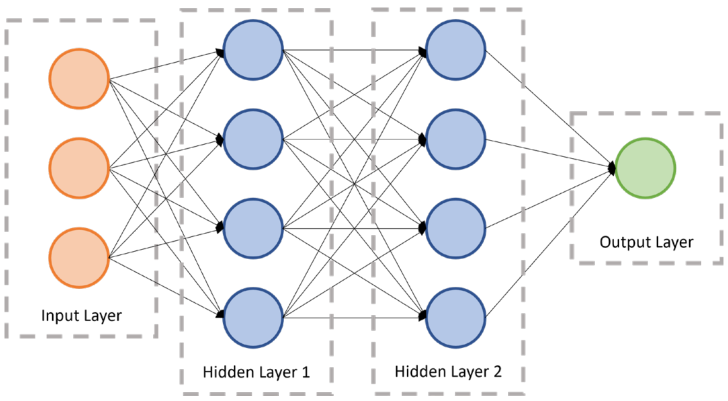


Figure 2.2 3-layer Neural Network

CNNs, first introduced by LeCun et al. (1989)[29], are a specialized type of neural network designed to process grid-like data, particularly images. Over the years, CNNs have achieved remarkable results in various computer vision tasks, such as object recognition, segmentation, and detection [30-32], and have become a cornerstone in the field of deep learning. To fully comprehend the architecture and functionality of this type of network, it is essential to identify its key properties and components. CNNs are comprised of three primary types of layers: convolutional layers, pooling layers, and fully connected layers.

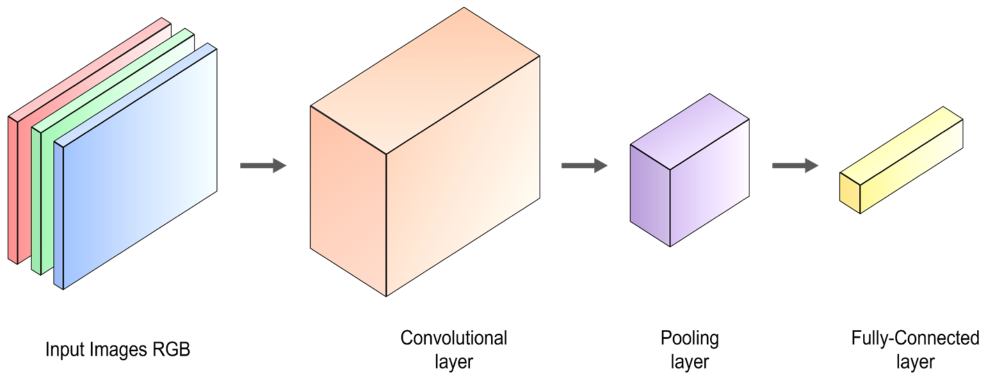


Figure 2.3 Example of a convolutional neural network

Input Images

The input image is represented as a matrix of pixels; therefore, it is necessary to define the size of these inputs. The size of an image is commonly determined by its height and width, which is known as resolution. However, since images are usually in colour, it is necessary to add another dimension: depth, also called channel size. A generic colour image is represented using the RGB convention (Figure 2.3). When processing RGB images with a CNN, the input layer typically has three channels, one for each colour (red, green, and blue). These channels are processed independently and in parallel by the convolutional layers, allowing the network to learn different features related to colour information. As the information flows through the network, the convolutional and pooling layers extract features from increasingly higher levels, which are ultimately used by the fully connected layers to make predictions, such as image classification or object detection.

Convolutional Layers

Convolutional layers are the foundation of CNNs and are responsible for extracting local features from input images (Figure 2.3). These layers use filters or convolutional kernels to perform convolution operations on specific image regions, allowing them to identify patterns such as edges, textures, and shapes.

The convolution operation is a mathematical function that combines two functions, in this case, an input image and a filter, to produce a third function that represents a modified version of the original image. Convolutional filters are matrices of weights that are applied to regions of the input image, allowing them to detect specific patterns. These filters can be learned automatically during the training process, enabling the CNN to adjust its parameters to maximize performance on a given task [33].

Padding and stride are two key concepts in the application of convolutions. Padding refers to the process of adding zeros around the input image to maintain spatial resolution after applying convolution (Figure 2.4). Stride, on the other hand, determines the distance between consecutive applications of the convolutional filter. These two parameters allow more precise control of the spatial resolution of the output and, consequently, the number of parameters and computational complexity of the network [34].

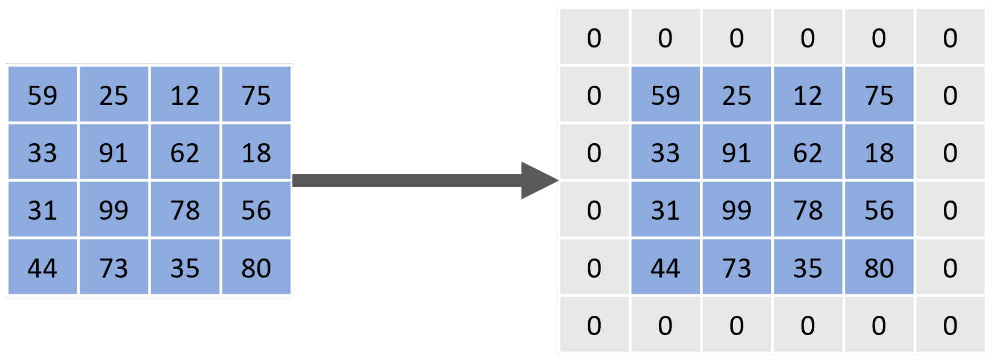


Figure 2.4 Zero-padded 4 x 4 matrix becomes a 6 x 6 matrix.

Pooling Layers

Pooling layers (Figure 2.3), also known as subsampling layers, aim to reduce the spatial dimensions of image representations, thereby decreasing the number of parameters and computational complexity of the network. Moreover, these layers provide some invariance to minor variations and transformations in images, enhancing the robustness of the CNN [35].

Several types of pooling operations exist, with max pooling and average pooling being the most prevalent. Max pooling selects the maximum value within a sliding window applied to the image representation (Figure 2.5), whereas average pooling computes the average of the values within the window. Both operations diminish the spatial dimensions of the representation; however, max pooling generally preserves more distinctive features than average pooling [36].

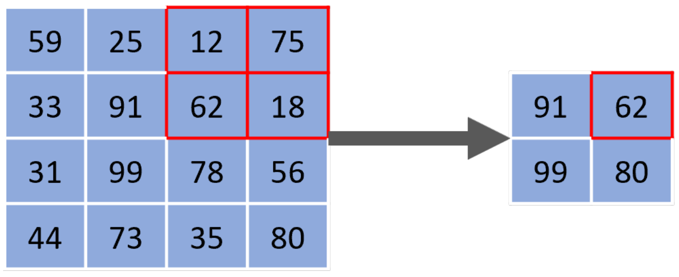


Figure 2.5 Max pooling operation.

Fully Connected Layers

Fully connected layers (Figure 2.3), also known as dense layers, are commonly found at the end of CNN architectures and aim to consolidate the information extracted in convolutional and pooling layers to perform classification or regression on the specific task.

These layers receive one-dimensional vectors, which are the result of flattening the multidimensional representations obtained in the previous layers.

Fully connected layers consist of neurons that apply non-linear activation functions to the inputs, allowing the CNN to learn complex relationships between the extracted features and target labels. Some of the most commonly used activation functions include ReLU (Rectified Linear Unit), sigmoid, and tanh (hyperbolic tangent). ReLU is a simple and computationally efficient activation function introduced by [37]. ReLu is a piecewise linear or nonlinear function with solid biological and mathematical foundations. Glorot et al. showed that ReLu enhances deep neural network training significantly [38]. The function is defined as:

$$f(x) = max(0, x) \tag{1}$$

ReLU outputs the input value itself when the input is positive and outputs 0 when the input is negative or zero. Figure 2.6a shows a visual representation of the ReLu function.

The sigmoid function, also known as the logistic function, the graph of the sigmoid function is an S-shaped curve as shown in Figure 2.6b. The function is defined as:

$$s(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

The sigmoid function outputs values between 0 and 1, making it particularly suitable for binary classification problems where the goal is to distinguish between two classes [39].

The hyperbolic tangent function, or tanh, is a scaled and shifted version of the sigmoid function [40]. It is defined as:

$$t(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})} \tag{3}$$

The tanh function outputs values between -1 and 1, which is shown in Figure 2.6c. It is often used in the hidden layers of neural networks because it is zero-centred, which can help improve convergence during training.

Convolutional, pooling, and fully connected layers must work together well for CNNs to be effective in solving computer vision challenges. With this combination, the network can learn feature hierarchies that range from basic components like edges and textures to more complex ones like objects and scenes.

These core elements have been improved and refined in more recent years by sophisticated CNN architectures. For instance, He et al. (2016) [32], Szegedy et al. (2015) [41], and Ronnberger et al. (2015) [42] used residual blocks, inception modules, and skip connections. These developments have made it easier to create deeper, more effective CNNs, which has resulted in previously unheard-of performance in tasks like object identification, semantic segmentation, and object detection.

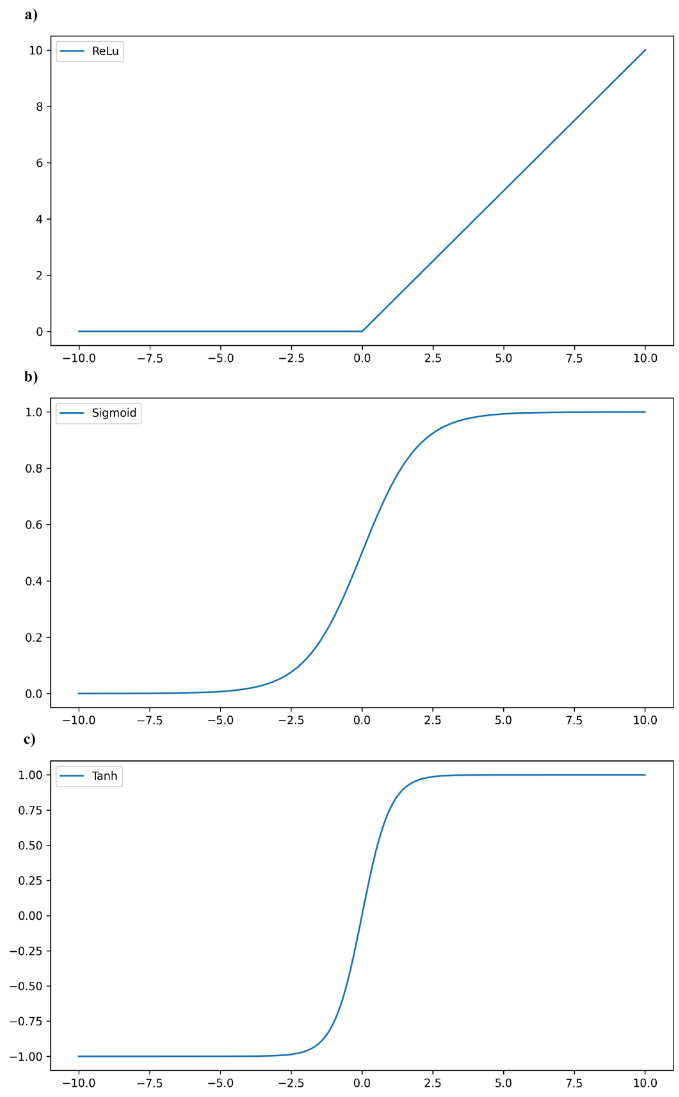


Figure 2.6 a) ReLu function; b) Sigmoid function; c) Tanh function

3. Literature review

Crack monitoring plays a crucial role in assessing the structural integrity and safety of civil engineering structures like roads, bridges, buildings, and pavements are frequently under high levels of physical stress, which may be caused by everyday use or natural catastrophes like earthquakes. The cracks weaken the component, lower its capacity for loading, and cause surface discontinuities [3,43,44]. The detection of cracks at an early stage can significantly reduce the cost of maintenance and prevent catastrophic failures [45].

There are several techniques utilized for crack monitoring. Destructive testing (DT) methods and non-destructive testing (NDT) methods are two major categories into which these techniques can be divided.

In order to find and evaluate the cracks, DT techniques involve removing a sample from the concrete structure and evaluating it in a lab. NDT techniques are often utilized in civil engineering to find cracks in infrastructure and buildings and to characterize them without endangering the building itself.

In this instance, non-destructive testing, ranging from manual detection methods to automated deep learning methods, will be used to analyze the overview of crack detection.

A list of the topics to be covered is presented, with a major focus on image-based techniques.

- Visual Inspection
- Ultrasonic Testing
- Acoustic Emission
- Ground-Penetrating Radar
- Terrestrial Laser Scanner
- Image-Based techniques

Visual Inspection

The most common method, manual visual inspection, is frequently used to detect cracks in civil engineering structures [46]. This technique involves a thorough inspection of the structure by an inspector equipped to detect any cracks that may be visible. However, it has many limitations, such as costly labour, subjectivity and accessibility, which are only available in some parts of the structure.

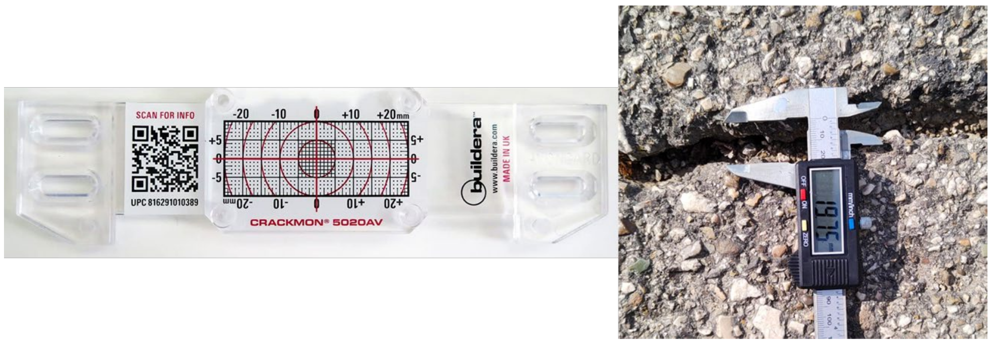


Figure 3.1 Instruments used for visual inspection of cracks. On the left tell tale, and on the right a calliper.

Ultrasonic Testing

Ultrasonic testing (UT) is a well-established method that uses high-frequency sound waves to detect cracks in concrete structures [47]. The use of pulse, pitchcatch, or through echo transmission techniques may be used for the purpose of conducting UT. It is capable of detecting surface and internal cracks as well as giving information on their depth and orientation [48].

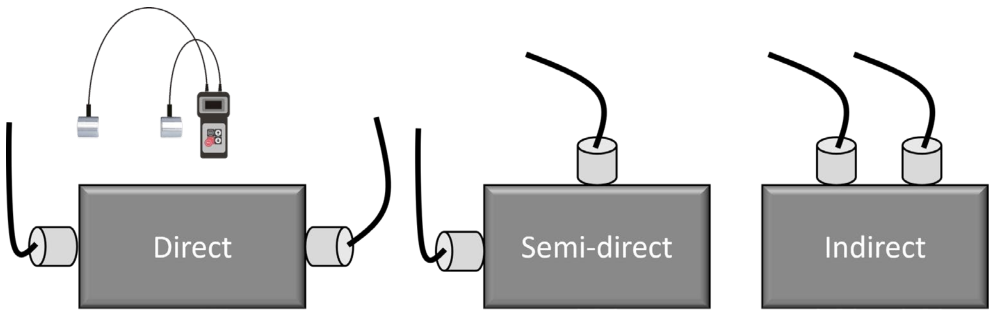


Figure 3.2 Instrument used for ultrasonic testing. Three ways of transmission.

Acoustic Emission

Acoustic emission (AE) is a passive method that monitors the release of energy in the form of elastic waves generated by the growth of cracks in concrete structures [49]. Active cracks may be found and located using AE in real-time, giving information on the damage's severity and the crack's growth [50]. AE is particularly sensitive to defect activity when a structure is loaded over its service load in a proof test, unlike ultrasonic testing, which actively probes the structure.

Ground-Penetrating Radar

Ground-penetrating radar (GPR) is a method that uses the propagation and reflection of electromagnetic waves to detect and locate cracks [51].

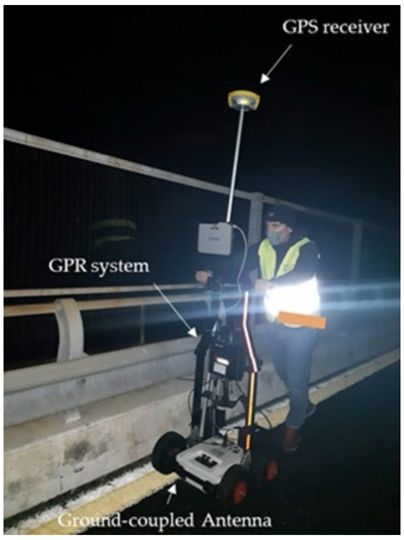


Figure 3.3 GPR System

GPR can be employed to detect and monitor surface and subsurface cracks in concrete structures [52]; also, GPR has been extensively used to identify and evaluate cracks and other defects in asphalt and concrete pavements, which can help inform maintenance and rehabilitation strategies [53].

Terrestrial Laser Scanner

Terrestrial Laser Scanning (TLS), also known as terrestrial LiDAR (Light Detection and Ranging) or topographic LiDAR, is an advanced remote sensing method that utilizes a laser source to project light pulses onto an object's surface [54]. This technique allows the XYZ coordinates of numerous points on the surface to be acquired [55]. The laser scanners used in civil engineering are those of the distance measurement type or also known as 'Ranging Scanners'. There are two principles by which distance is measured: systems that are based on the measurement of time of flight (TOF, Time of Flight) and systems that are based on the measurement of the phase difference of the sent and reflected signal (PS, Phase Shift).

The time it takes for a laser pulse to travel from the scanner to the item and back to the scanner is what is measured by the ToF technique. The scanner can utilize this knowledge to determine the object's distance because the speed of light is constant. The accuracy of time of flight scanners can be affected by atmospheric conditions and the reflectivity of the target surface, and they often have a slower acquisition rate than phase shift scanners. However, they can estimate distances across extended distances (a few kilometres). By precisely calculating the time-of-flight and considering the scanner's orientation, the three-dimensional (3D) coordinates of the object's surface can be ascertained, resulting in a comprehensive point cloud that embodies the object's geometric features [56].

Phase shift scanners emit a continuous wave of light and measure the phase shift, or change in the wave's cycle, when the light wave returns to the scanner after being reflected off the object. The light's path distance can be determined from this phase change. At short to medium ranges (up to a few hundred meters), phase shift scanners are more precise than ToF scanners and can capture data at very high rates (millions of points per second). However, they could have trouble seeing farther out, and they might be more sensitive to interference from ambient light. Figure 3.4 shows the basic types of terrestrial 3D laser scanners.

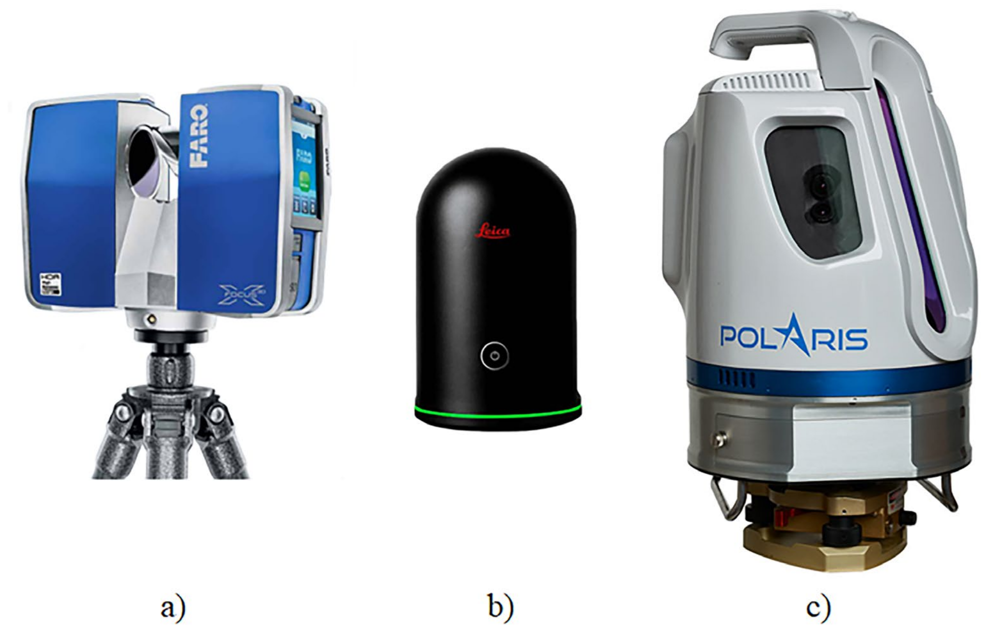


Figure 3.4 Three TLS: a) Faro Focus X 330; b) Leica BLK360; c) Polaris Teledyne Optech

This technology is in a phase of continuous development, and several improvements are being made over time in order to achieve increasing measurement accuracy and acquisition speed. In addition to the 3D coordinates of the object points, the sensors allow the power of the reflected pulse to be determined. Some systems allow echo-digitalisation of the returning laser signal and automatic classification by means of 'full waveform' analysis [57].

TLS has gained widespread acceptance for detecting cracks in a diverse array of civil structures, encompassing bridges [58], buildings [59], and roads [60]. Through the processing and examination of point cloud data acquired via TLS, experts can discern the existence, location, and dimensions of cracks, providing valuable insights for assessment and repair. In most cases, inspections are based on a visual and highly subjective interpretation to classify surface conditions [61].

Terrestrial Mobile Laser Scanners, frequently referred to as Mobile Laser Scanners (MLS), are particularly captivating due to their unique capabilities. These systems enable the capture of 3D data through one or more scanners mounted on various mobile platforms, such as vehicles, boats, and trains [62].

Additionally, some market-available solutions incorporate cameras for enhanced functionality. When employed for pavement surface scanning, MLS systems typically offer 3D accuracies of 1 centimetre and a point density exceeding one thousand points per square meter [1]. Guan et al. developed an evaluation methodology for crack characterisation based on MSTV (Multi-scale Tensor Voting) [63].

Despite its advantages, this technology also presents some limitations. In addition to higher operational costs, it also presents challenges related to technical performance under specific conditions, including:

- Certain parts of the structure may be invisible to the laser scanner (e.g., the bottom parts of the bridge deck)
- The texture of the examined structure components may not be accurately captured, potentially concealing any corrosion of the elements
- TLS can produce extensive amounts of data, much of which is redundant, which complicates the data processing task.

Image-Based techniques

Image-based techniques can be a powerful alternative for remotely monitoring and inspecting civil structures. By collecting pictures of a specific structure from various perspectives with cameras [64-66], robotic image acquisition systems [67], or unmanned aerial vehicles (UAVs) equipped with high-resolution cameras, visual information can be collected [68,69].

Robust autonomous image processing techniques are used to evaluate the collected images and search for any cracks after acquiring the necessary visual data.

A variety of approaches have been employed in the identification of image cracks. Initial investigations in this domain utilized methodologies that did not depend on data-driven learning but rather hinged upon the application of bespoke computer vision attributes. Such studies encompassed the implementation of techniques such as thresholding, mathematical morphology, Gabor filters, the Otsu method, wavelets, and edge detection methods, including the Canny technique.

N. Tanaka and K. Uematsu (1998) suggested a technique with four steps—black pixel extraction, saddle point detection, linear feature extraction, and contouring processing. While saddle point identification uses a top-hat transform operation to increase dark regions surrounded by light regions, which match the linear characteristics of cracks, black pixel extraction uses local minima in the picture to discover black pixels. The technique requires specifying particular settings and presupposes that the cracks' area is not very vast in comparison to the background [70].

Iyer et al. (2005) developed a three-step method for crack detection in high-contrast images. Their approach combines curvature evaluation and mathematical morphology techniques to detect crack-like patterns in noisy environments. The segmentation process defines the crack-like pattern with respect to a precise geometric model. After cross-curvature evaluation, linear filtering was performed to distinguish these patterns from similar background features. By using geometry-based recognition systems, they were able to sequentially identify irregularities in the images related to crack features [71].

Yiyang et al. (2014) proposed a crack detection algorithm utilizing digital image processing technology. To extract information from images containing cracks, the researchers carried out preprocessing, image segmentation, and feature extraction. A threshold segmentation method was applied after smoothing the image. The authors calculated the images' area and perimeter, as well as a roundness index, to evaluate them. By comparing these indices, they were able to determine the presence of a crack in an image [72].

AMA Talab et al. (2016) used the Otsu method and multiple filtering in image processing techniques to find cracks in concrete structures. The Otsu method is a thresholding approach that effectively determines the best threshold value for distinguishing the crack from the background. It divides a picture into foreground and background. In order to find cracks in images of concrete buildings, the paper introduces a novel image-processing technique [73].

Salman et al. (2013) proposed an automatic crack detection approach for digital images using Gabor filtering, which is highly effective in multidirectional crack detection for pavement images with rich textures. The image analysis of the Gabor filter function directly relates to manual visual perception. The preprocessed pavement image is convolved with the filter, and the real component of the resulting image is thresholded to generate a binary image. Next, binary images from differently oriented filters are combined to produce an output image containing detected crack segments. After completing the filtering process, cracks aligned in various directions are detected. Their proposed methodology achieved a detection precision of 95% [74].

Huang et al. (2006) introduced a classification approach in which images are initially divided into 8x8 pixel cells, and the cells are then categorized as either crack or non-crack cells. To more effectively evaluate the method's accuracy, they directly processed each pixel, classifying it as a crack or non-crack seed based on the contrast between each element and its neighbouring pixels. They employed eight templates to gauge the contrast in various directions. If the contrast values surpassed a predetermined threshold, the pixel was recognized as a crack seed and labelled as diagonal, transverse, or longitudinal, depending on the highest contrast value. Following that, nearby crack seeds were linked together. Crack pixels shorter than a specific length were considered noise and removed. Although this method is relatively quick, determining universal thresholds that apply to images with diverse contrast, lighting, and shadow effects is difficult. Consequently, it might not be sufficiently robust for the quality of the images supplied [75].

Huili Zhao et al. (2010) present an improved Canny edge detection algorithm (Canny 1986) for pavement edge detection applications, addressing issues like weak edge detection and grayscale distinction. The algorithm employs the Mallat wavelet transform and quadratic optimization of the genetic algorithm to enhance edge detection. The new model meets real-time pavement edge detection requirements, compensating for Canny algorithm drawbacks, effectively detecting pavement image edges, and reducing processing time while eliminating noise and preserving unclear edges [76].

Abdel-Qader et al. (2013) discovered that the Haar Wavelet technique was the most reliable edge detection approach when comparing various edge detection algorithms. However, the performance of edge detection algorithms, as well as morphological operation-based methods, on noisy image data remains a matter of debate [77].

Traditional computer vision (CV) techniques often struggle with weak generalization and adaptation abilities, as well as a heavy reliance on the quality of the images. Moreover, the complex environment of surfaces can lead to poor camera settings, resulting in issues such as low contrast, inconsistent lighting, and significant noise. These factors make it challenging to build an efficient detection model using conventional CV approaches [78].

However, significant advancements in crack identification have been made thanks to ML and DL techniques, which have proven highly successful in semantic segmentation. The goals of crack segmentation align well with the semantic segmentation technique, which predicts a classification label for each pixel. Convolutional neural networks (CNNs), a key subfield in DL, have demonstrated promising results in pixel-level detection of target objects within noisy images [6,7].

The following paragraphs provide a comprehensive overview of various studies that utilize machine learning methods in conjunction with segmentation and feature extraction techniques. Additionally, it explores studies that specifically utilize CNNs.

Lattanzi et al. (2014), to improve accuracy and speed in crack identification under diverse environmental circumstances, created an artificial clustering approach for segmentation based on Canny and K-Means. Their work is particularly significant as it contains images from several locations, as is often the case with real-world bridges. It is crucial to take into account the environmental variability caused by different lighting and shading conditions at different bridge locations. The study focused on the effectiveness of the image segmentation and feature extraction phases. They compared the Canny edge detector and Haar wavelet transformation to the K-Means strategy, which builds clusters in relation to the mean value of clusters. Naive Bayes, decision trees, and k-NN were used to evaluate the segmentation results, while Bayesian networks, decision trees, neural networks, and k-NN were used to assess the overall classification performance. For the image variance test, Naive Bayes results and the Canny edge operator outperformed other classifier segmentation. However, decision tree and k-NN scores outperformed Naive Bayes results in terms of segmentation effectiveness [79].

Zhang et al. (2014) used the k-NN, support vector machine, radial basis function neural network, and extreme learning machine in their 2014 study to classify cracks in subway tunnels. For image segmentation, they used a number of approaches, including average smoothing, black top hat transformation, and thresholding. Additionally, they used statistical feature extraction techniques, focusing on standard deviation from shape distance histograms. While the test accuracies of the classifiers were comparable, the extreme learning machine achieved the top score of 91.6%, followed by the support vector machine, radial basis function neural network, and k-NN classifiers. The k-NN classifier scored 88.7% in this experiment [80].

Moussa and Hussain (2011) introduced a method for automatic crack detection, classification, and parameter estimation using machine learning. In order to distinguish between the background and crack pixels in an image, they used Graph Cut segmentation. A binary vector is made after segmentation. Seven features are extracted from the vector for classification purposes. The crack type is then classified as transverse cracking, longitudinal cracking, block cracking, or alligator cracking using a support vector machine. Additionally, Moussa and Hussain proposed a method for estimating the size and severity of a crack based on the crack's width and length in the image [81].

Prasanna et al. (2014) conducted crack detection on bridge images captured using robotic imaging and proposed a method called spatially tuned robust multi-feature (STRUM). SVM, Adaboost, and random forest were some of the machine learning techniques used to categorize pixels into crack and non-crack categories. Despite the noise in the images, they used robust curve fitting to find possible crack locations. When employing conventional image processing techniques, the accuracy was just 69%; however, the STRUM classification strategy reached 95% accuracy. The performance of this method on real bridge data showed that accuracy could be increased. The classifier could be built using training data from multiple locations on the bridge, and testing could be done using data from a different location with a comparable structural composition [82].

Yang et al. (2008) employed wavelet transformation and the co-occurrence matrix for image segmentation and feature extraction. A support vector machine (SVM) and two neural network-based classifiers, the back-propagation neural network and the radial basis network, were used to train the obtained features. The SVM with a radial kernel was observed to perform better than the back-propagation neural network, radial basis network, and SVM with a polynomial kernel [83].

Varadharajan et al. (2014) used a selective method for image-based analysis, concentrating on images taken during the day under mostly cloudy conditions to ensure good lighting. This approach has a disadvantage in that because of the laborious selection procedure; all collected images must be stored before being picked out and analyzed, which results in a large amount of data storage. The researchers also used machine learning in their study, taking into consideration input images with background objects like vehicles, traffic signs, and buildings. The ground plane was separated from the rest of the picture in the first phase, and then feature descriptors were computed using the colour and texture of the preprocessed pixels. Utilizing nine features and data collected from human annotators, a support vector machine was effectively trained to classify the images [84].

Li et al. (2014) divided images into crack regions and non-crack regions by calculating the difference between the maximum and minimum grayscale values within an image area. Subsequently, the foreground was distinguished from the background using Otsu's segmentation method. Finally, the images were classified using binary trees and back-propagation neural networks to achieve a comprehensive analysis [85].

Moon et al. (2011) employed a series of preprocessing methods, including median subtraction, Gaussian low-pass filtering, threshold segmentation, and morphological operations for feature extraction. Cracks were differentiated from the background image through filtering, an improved subtraction method, and morphological operations. A neural network was then utilized to automate image classification. The algorithm was tested on real surface images of a concrete bridge and demonstrated high accuracy in image classification. However, the test faced limitations due to similar environmental conditions, indicating that the proposed algorithm should be evaluated across various fields of application [86].

Recent developments based on deep convolutional neural networks have demonstrated a more effective approach to automated crack detection. CNNs networks function differently from other machine learning classifiers, as the framework inherently includes a feature extraction step and does not require image segmentation as a preprocessing step. The benefit of CNNs is their end-to-end segmentation method, which considerably reduces the need for human involvement in the segmentation process. Compared to manually crafted features used in traditional approaches, the features obtained by CNNs offer superior performance. Leveraging this deep learning capability, several efforts have been devoted to developing reliable feature representations for segmenting crack images. Deep neural networks utilize characteristics to determine the presence of cracks in image patches by combining pixel-level classification confidence from multiple frames with varying lighting conditions.

Zhang et al. (2016) proposed a supervised deep CNN to classify each image patch (99x99x3) in the collected images, with features automatically learned from manually annotated patches obtained using a low-cost sensor, such as a smartphone. The proposed method demonstrates superior performance in accurately classifying crack patches from background ones when compared to existing handcrafted methods [3].

Allen Zhang et al. (2017) introduced CrackNet, a five-layer CNN architecture composed of two convolutional, two fully connected and one output layer for pixel-level crack identification on 3D asphalt surfaces. Although CrackNet was effective in accurately detecting pixel-level cracks, the processing time of the architecture proved to be significant [87].

Gopalakrishnan et al. (2017) proposed various image classifiers for detecting cracks in Hot-Mix Asphalt (HMA) and Portland Cement Concrete (PCC) by transferring features learned from ImageNet [30]. They concentrated on a transfer learning strategy and optimized pre-trained networks for the crack-detecting task. For pavement distress detection, VGG-16, a pre-trained CNN, was used. Seven hundred sixty photos were used to train the network, while 212 images were used to test it. The classifier layers of CNN were swapped out for random forest, very randomized trees, SVM, and logistic regression classifiers in order to conduct a comparative study. The pre-trained network's original version received the highest score in the research with a claimed 90% accuracy, making it the best choice [88].

Haifeng Li et al. (2018) extracted potential cracks at each scale in the image using a technique based on the lowest intensity path of the window, evaluated the relationships between the cracks at various scales, and created a crack evaluation model based on multivariate statistical hypothesis [89].

Tong et al. (2018) developed a two-stage CNN-based model also to detect asphalt pavement crack length. The results presented show that the training strategy used produces an increase in accuracy [90].

Fan et al. (2018) trained a CNN model as a multi-label classifier for pavement crack detection, implementing a strategy that involved modifying the ratio of positive to negative samples. The researchers utilized CNN to detect pavement cracks from images captured by an iPhone on pavements in Beijing, China. The proposed methodology showcased a precision of approximately 92%, surpassing traditional machine learning techniques. The results indicated that the approach effectively managed varying pavement textures and successfully recognized different pavement conditions [91].

Nguyen et al. (2018) used a U-Net-based architecture for semantic pixel-wise segmentation of road and pavement surface cracks. For model training and validation, the authors employed both publicly available and their own privately collected crack datasets; however, the model's crack detection accuracy needs improvement [92].

Lee et al. (2019) used as crack detection network a CNN with trainable up-sampling layers for segmenting cracks. The network was built on pixel-wise classification utilizing information from both full photos and patch images, as well as image segmentation using the MS-COCO [93] database and a pre-trained approach. The approach highlighted the lack of data available for semantic segmentation by using just 242 real images as a database. More pictures were produced utilizing a synthesizing technique based on a 2D Gaussian kernel and Brownian motion to overcome this problem [94].

Baoxian Li et al. (2020) proposed a CNNs used to classify cracks patches into five categories based on 3D pavement images. They used WayLink's PaveVision3D Ultra to acquire 3D images [95].

Fan et al. (2020) demonstrated the use of Deep CNNs to detect and recognize cracks as defects with quantifiable properties in applications for crack detection on pavement surfaces (e.g., crack length and size). In a separate paper, the authors propose a modified version of U-Net in which two modules were added to the overall architecture to increase the performance of crack segmentation: dilation, convolution and hierarchical feature learning module [96,97].

Guo et al. (2021) introduced BARNet, an encoder-decoder network developed for crack detection. Base Predictor Module, Edge Adaptation Module, and Refinement Module are the three essential parts of this network. Crack localization's problem with erroneous borders is successfully resolved by BARNet [98].

Liu et al. (2021) proposed a crack transformer encoder-decoder structure called CrackFormer. It includes a self-attention block and a scaling-attention block for fine-grained crack identification. By using a cascaded self-attention module to collect feature dependencies across large distances, these transformer-based approaches are able to gain better global information [99].

Gui Yu et al. (2022) proposed a U-shaped encoder–decoder semantic segmentation network combining U-Net with ResNet and used the scSE attention module to enhance the crack, which is called RUC-Net [100].

In conclusion, the development of crack detection algorithms is an ongoing process driven by continuous technological advancements. While pixel-based image processing remains a prevalent technique, more sophisticated neural network implementations are emerging as competitive and expedited alternatives. Each crack detection algorithm examined herein possesses certain limitations, with gaps at various stages and issues that warrant further attention. These areas of concern are identified as crucial avenues for subsequent research.

The present thesis introduces a methodology for semantic crack image segmentation predicated on the residual structure, which is constructed upon the U-Net model's architecture, utilizing a ResNet-50 encoder. This approach circumvents the necessity for pre-identification of the region of interest, facilitating the automatic segmentation of cracks in images with intricate backgrounds. Furthermore, it enables the independent learning of crack characteristics and the extraction of supplementary feature information.

4. U-Net-Based Crack Segmentation

The methodology we proposed aims at the segmentation of cracks using Deep Learning CNN techniques from images acquired with commercial cameras and implemented in a Python environment.

The data set used is the one proposed by Yang et al. [101]; the model implemented uses a modified U-Net to improve the automatic crack segmentation process. The proposed methodology is based on the following steps:

- Use of the Crack500 Dataset [101]
- Data Augmentation on the Dataset
- Implementation of the U-Net Model with ResNet50 encoder pretrained with ImageNet
- Model training
- Metrics analysis

A comparison will be made with the main results reported in the literature, in particular with those obtained by Lau et al. [102].

Figure 4.1 shows the workflow of the proposed methodology.

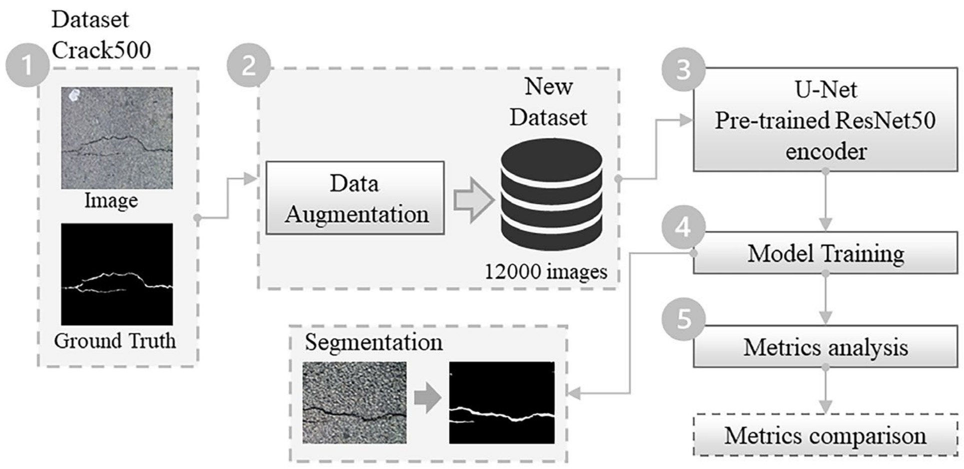


Figure 4.1 Crack segmentation model WorkFlow

4.1 Dataset

The Crack500 dataset was proposed by Yang et al. [101] and contains images acquired using mobile phones at the main campus of Temple University (Philadelphia, USA).

It initially consisted of 500 images of diverse sizes, around 2000 × 1500 pixels. Each pavement crack image has a pixel-level annotated binary map. It was divided into 250 training samples, 200 tests and 50 validation samples.

To make the training process more efficient, a set of data augmentation procedures was run on the data, a technique that has been developed to reduce the overfitting [103].

Using Python, each image is cropped into six image regions overlapped by 80 pixels, flipped horizontally and then rotated in 90-degree steps, starting from 0 degrees up to 270 degrees (Figure 4.2). As a result, all the images and the corresponding masks with a size of 512 x 512 pixels were obtained.

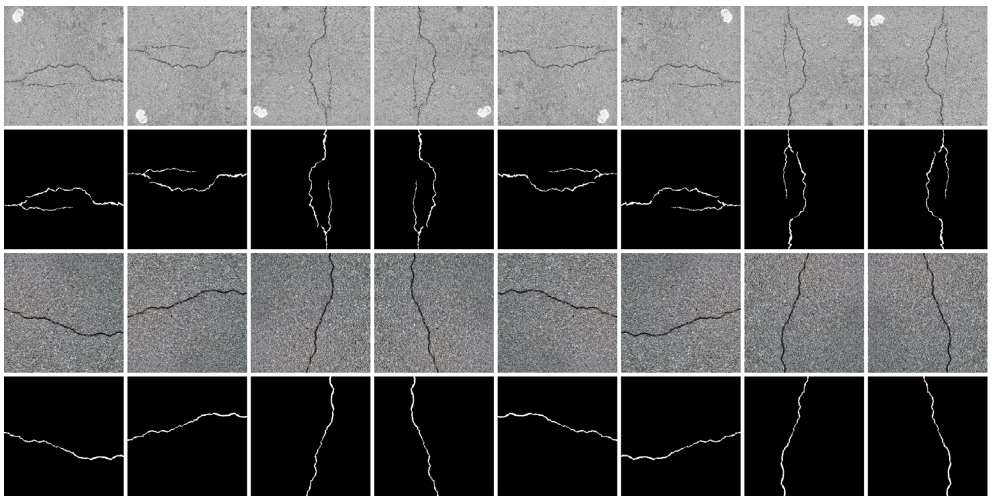


Figure 4.2 Example of Dataset and Data Augmentation.

4.2 Network architecture

The network architecture proposed is a U-Net-based architecture with a ResNet50 encoder. The U-Net was presented by Ronneberger et al. [42] at the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI) in 2015. In their work, Ronneberger et al. demonstrated that U-Net could outperform previous state-of-the-art techniques on a range of biomedical image segmentation tasks.

The U-Net architecture is named after its characteristic U-shaped structure. This deep learning model consists of an encoding (downsampling) path and a decoding (upsampling) path. The encoding path uses convolutional layers to extract and condense information about the input images, while the decoding path uses transposed convolutions to output precise segmentation maps. A unique feature of U-Net is the presence of skip connections between corresponding layers in the encoder and decoder parts, which helps in preserving spatial information lost during downsampling. The architecture of the U-net is explained in Figure 4.3.

A ResNet50 encoder trained on the ImageNet dataset [30] is utilized in this encoder-decoder-based design. The model quickly converges thanks to the usage of a pre-trained encoder. The input image is passed into the pre-trained ResNet50 encoder, whose fundamental building block is a set of residual blocks (Figure 4.4).

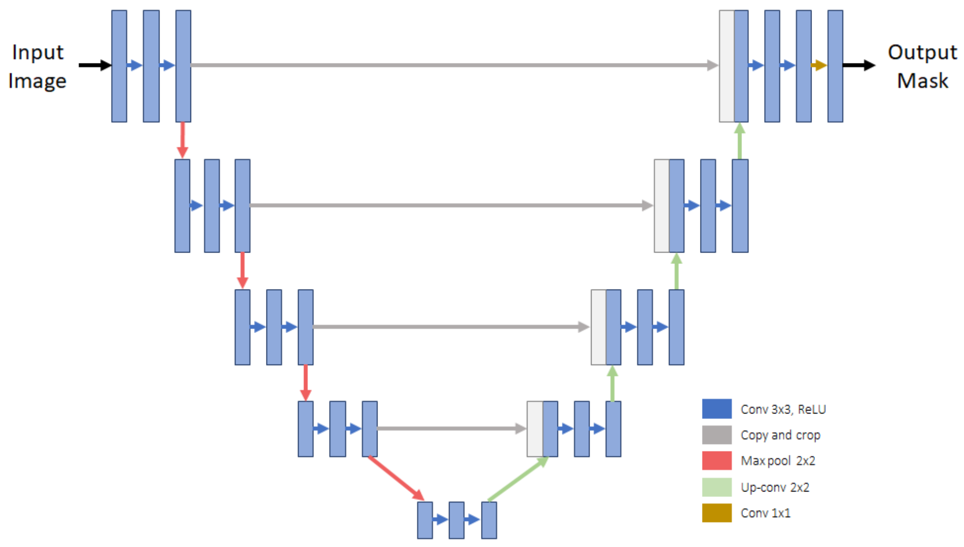


Figure 4.3 U-net.

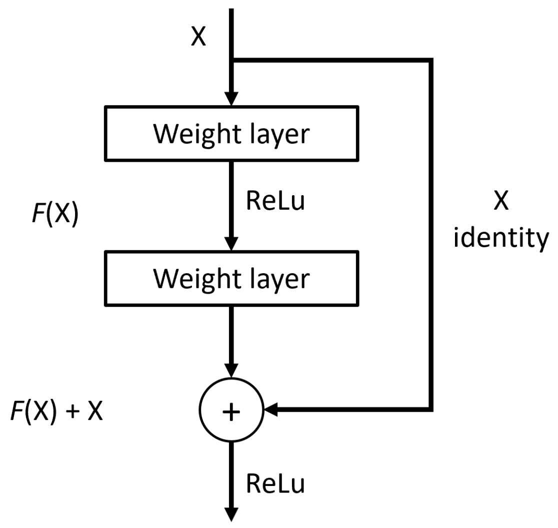


Figure 4.4 Example of single residual block

The relevant features from the input images are extracted by the encoder with the assistance of these residual blocks, and these characteristics are then sent to the decoder. A transpose convolution is started by the decoder to upscale the input feature maps into the proper form. The specified shape feature maps from the pre-trained encoder are then concatenated with these upscaled feature maps using skip connections. By assisting the model in obtaining all the low-level semantic data from the encoder, these skip connections enable the decoder to produce the necessary feature maps. The two 3x3 convolution layers are then added after that, with a batch normalization layer and a ReLU non-linearity layer coming after each layer. The output of the final decoder block is sent into a 1x1 convolution layer, which is then fed into a sigmoid activation function to produce the appropriate binary mask. The architecture of the proposed algorithm is shown in Figure 4.5.

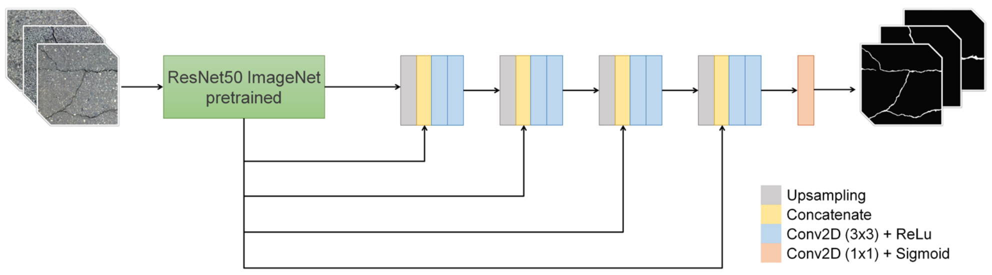


Figure 4.5 Implemented model.

The model uses Adam optimizer [104] with an initial learning rate set to 0.0001 and reduced by a factor of 0.1 in every 4 epochs, and cross-entropy loss as its loss function. We implemented the network in Python using Tensorflow/Keras. The workstation specifications used to train the neural network are TITAN X GPU (12 GB VRAM), Intel Core i7 processor, and 32 GB RAM.

In our work, the net was trained by using the training pair D = x(i), y(i), where x(i) is the i-th image patch and y(i) 0, 1 is the corresponding class label.

4.3 Model Evaluation Metrics

Two common evaluation metrics are utilized to assess the suggested approach in order to objectively estimate the performance of the network model. F1 score and Intersection over Union (IoU) are the conventional quantitative evaluation metrics utilized in our research. F1 is the combination of Precision and Recall and is computed as the harmonic mean of the two quantities [105]. Precision (P) is the proportion of correctly classified observations per predicted class, Recall (R) or Sensitivity is used to measure the percentage of actual positives which are correctly identified. Often, there is an inverse relationship between Precision and Recall: when precision increases, model sensitivity worsens and vice versa. For these reasons, it is important to find the golden mean, meaning a balance between the two indicators, to obtain a model that best fits the input data. The formulas used are:

$$F_1 = \frac{2PR}{P+R}; P = \frac{TP}{TP+FP}; R = \frac{TP}{TP+FN} \tag{1}$$

where TP are the True Positive (samples correctly classified as positive), FP are the False Positive (samples incorrectly classified as positive) and FN are the False Negative (samples incorrectly classified as negative). We do not consider the transitional areas (0 pixels distance) between non-crack and crack pixels. A detected crack pixel is considered a true positive if it is located 0 pixels away from the manually annotated crack.

F1 score is a combination of Precision and Recall and is a robust indicator for both balanced and unbalanced data sets. In general, F1 values greater than 0.9 are indicative of a very accurate classification; below 0.5, the classification may be considered inaccurate and, therefore unsuitable. Analysis of F1 is necessary when a balance between Precision and Recall is desired.

Intersection over Union (IoU) is a geometric type of evaluation metric. It describes the closeness of the predicted results to the ground truth bounding boxes and is expressed as:

$$IoU = \frac{Area(B_p \cap B_g)}{Area(B_p \cup B_g)} \tag{2}$$

Where B_p is the predicted bounding box, and B_g is the ground truth bounding box. In this case, the predicted bounding box represents the mask obtained with the proposed model (prediction output), and vice versa, the ground truth bounding box represents the mask used to train the implemented model (target mask).

Thus, the overlap occurs between the two masks, the predicted mask (prediction output) and the original mask (target mask). This means that IoU is equal to the number of pixels that are common between the target mask and prediction output divided by the total number of pixels in both masks. The higher the overlap, the higher will be the score; values close to 1 indicate an excellent overlap, and values below 0.5 indicate a poor overlap. In other words, should the predicted mask be identical to the mask used for training, the IoU would be 1. These metrics are commonly used in crack detection, but do not consider the subjectivity of manually labelled ground truth [106].

4.4 Results of U-Net Based Model

The trained model was applied to the Crack500 dataset. The original dataset consisted of 250 images of training data, 50 images of validation data, and 200 images of test data, and then the dataset was artificially increased with data augmentation technique to improve performance during the training phase. A total of 12000 images were used for training, 2400 images were used for validation and 9600 images for test. Figure 4.6 shows that both accuracy curves, the training and the validation set, increase and stabilize at high and similar values with a gap of 0.006, indicating that the model is learning correctly and generalizing well to unseen data.

Our model has a total of 20.6 million parameters, and the number of FLOPS is 236G; the inference speed on our hardware is 6.7 frames per second on images of 512x512. As a result, this strategy might not be advantageous for real-time applications and looks more suited for batch processing.

The metrics used for evaluating the proposed model are Precision (P), Recall (R), F1 and IoU, as described in Model Evaluation Metrics.

Table 1 shows the results obtained by applying the model trained with the proposed methodology and the results obtained by using the models re-implemented by both Lau et al. [102] and others [92,100] that achieve high enough and comparable accuracies.

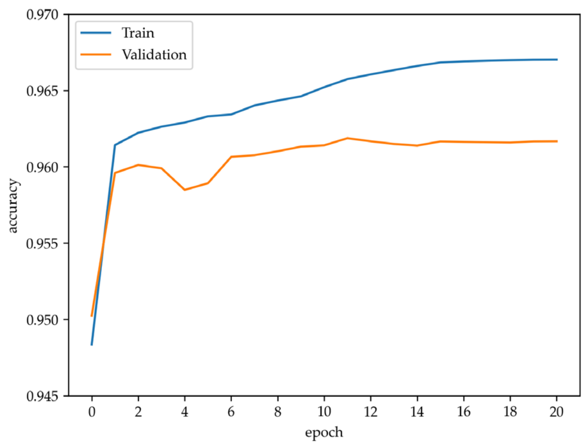


Figure 4.6 The trend of the accuracy curve during training. The epochs are on the X-axis, and the Y-axis is the prediction accuracy. In blue is the training curve, and in orange is the validation curve.

Table 1. Testing results of our model compared with other U-Net-based models on the same crack500 dataset.

Method	P	R	F1	IoU
U-Net by Nguyen	0.6954	0.6744	0.6895	0.5261
U-Net by Yu	0.6988	0.7619	0.7290	0.5736
U-Net by Lau	0.7426	0.7285	0.7327	0.5782
Proposed U-Net	0.8534	0.6813	0.7577	0.6248

It should be noted that the results listed in Table 1 are derived from the application of the same U-Net architecture on the same training Dataset (Crack500). Comparing the results shown in the table, those of Lau et al. emerge as the most accurate, resulting as the method working best at present and thus used by us as a benchmark.

Compared to the results of Lau et al., our model produced an increase in Precision, a slight decrease in Recall and an increase in F1 and IoU. The increase in Precision means that our model is more reliable, i.e., there are few false positives. However, the model may not predict all events by being less selective or sensitive (low Recall), i.e., false negatives may be many even if the model is accurate.

Lau et al. implemented a model with the goal of balancing Precision and Recall, since it is valuable that the model be precise, but it is also important to have adequate sensitivity (Recall). For us, the increase in Precision and the slight decrease in Recall still produced a balanced result, as can be seen by looking at the F1 score, which is close to 0.76.

The increase in IoU compared to the model of Lau et al. means that the predicted masks are more similar to the original ones (target masks). This is important when estimating crack widths; we assume that the masks used as training datasets are consistent with reality and, consequently, with crack width. Crack width is the main parameter used to derive severity levels [1].

Figure 4.7 shows examples of the results of the model trained on images referring to some major crack types. The images given as input to the model, shown in columns a, a1) belong to the test data set but are definitely not the ones used to train the model, which are others not being shown. Columns c, c1) show the results we obtained on those input images, while columns b and b1) show the hand-drawn masks available in the dataset for visual comparison with our outputs. Please note that in all cases, our output is more accurate than the hand-drawn masks. In detail, panels in column “a” show: (1–3) transverse cracking, (4–6) longitudinal cracking, and (7–9) block cracking, while panels in column “a1” show: (1–3) portions of alligator cracking, (4–5) edge cracking, and (7–9) portions of non-cracked pavement.

It should be pointed out that the portions of the pavement shown in Figure 4.7 belong to different types of wear layers, distinguished by different levels of adherence. Indeed, the images display very heterogeneous colour scales, sometimes marked by the presence of very evident stains due to the presence of aggregates of different types. Columns b and b1 display the target masks used to train the model, and columns c and c1 display the masks obtained by applying the trained model, the predicted masks.

In almost all cases, the masks predicted are better than those hand-drawn, in particular concerning the crack width, i.e., in row 5, panel c1(5), the crack was better delineated than the target mask shown in panel b1(5), which is less compliant, in terms of width, than the real configuration, panel a1(5). This aspect is crucial since, in the design of the maintenance plans, the main parameter regulating the severity levels of the different cracks is their width, in addition to the linear development and the area of extension [107].

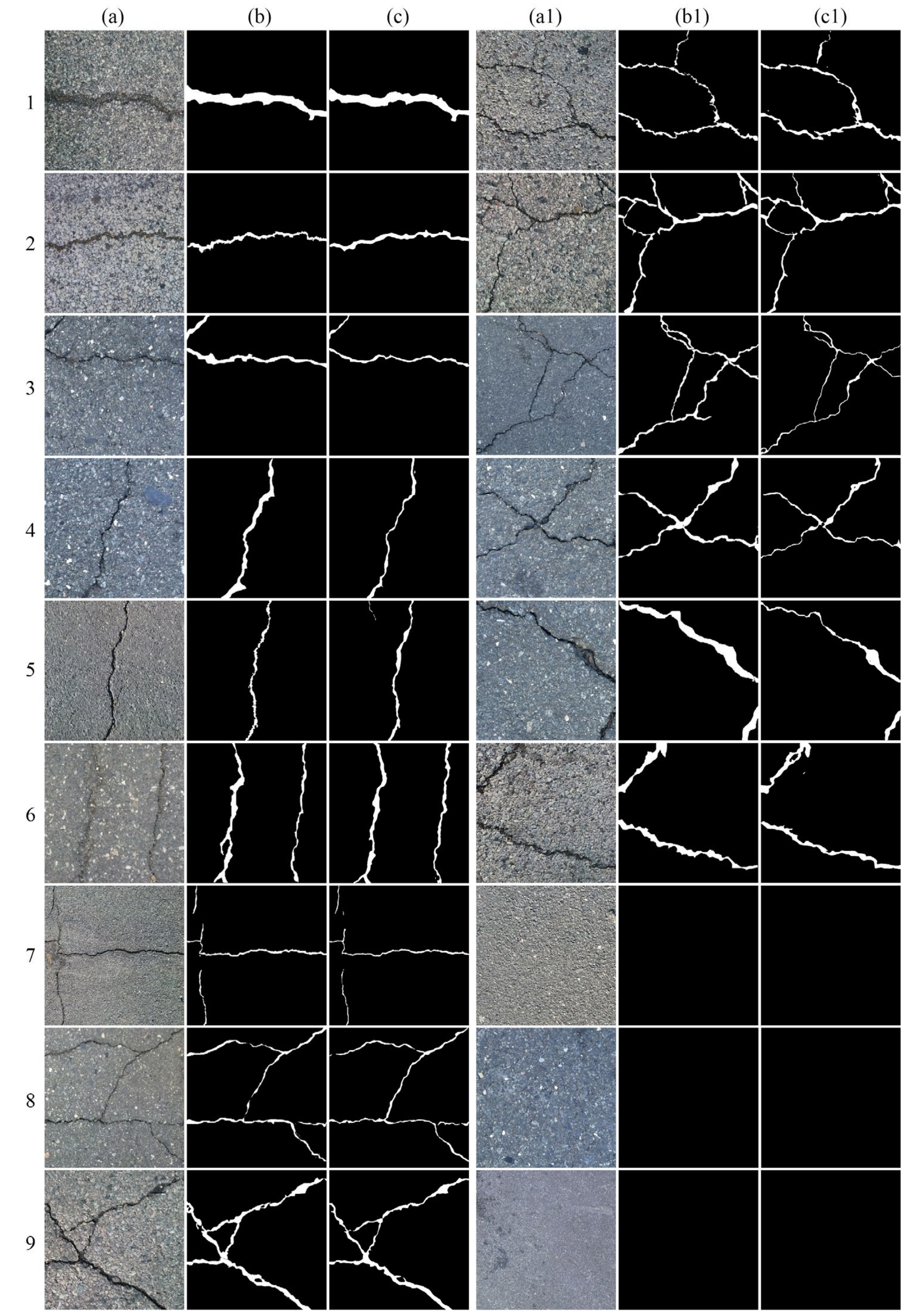


Figure 4.7 Output of the implemented model; (a, a1) Input images, (b, b1) Target masks, (c, c1) Predicted masks by our U-net

5. Methods and Applications in case studies

In this chapter, we begin by presenting a workflow of the methodology proposed in Section 5.1. This section explains the method used to filter a point cloud, measure and reconstruct cracks in 3D.

In Section 5.2, we present the case studies and results of the U-Net-based model applied to the images of real cases. This includes the filtering of point clouds from these case studies, verification of crack measurements, and results from associating the image with the point cloud.

5.1 Methods

The overall methodology is illustrated in Figure 5.1 and consists of three macro-phases, two of which make use of the U-Net-based model proposed in Chapter 4.

The process begins with the photogrammetric phase, using two real case studies. The tie point cloud is then filtered by an innovative ML method to improve the accuracy of the camera parameters, and subsequently generate the dense point cloud.

In the second phase, metric measurements of the cracks are carried out by analyzing the masks obtained as a result of crack segmentation of an orthophoto.

Finally, the third phase involves associating the cracks with points in the dense cloud and adding a label to these points to facilitate analysis.

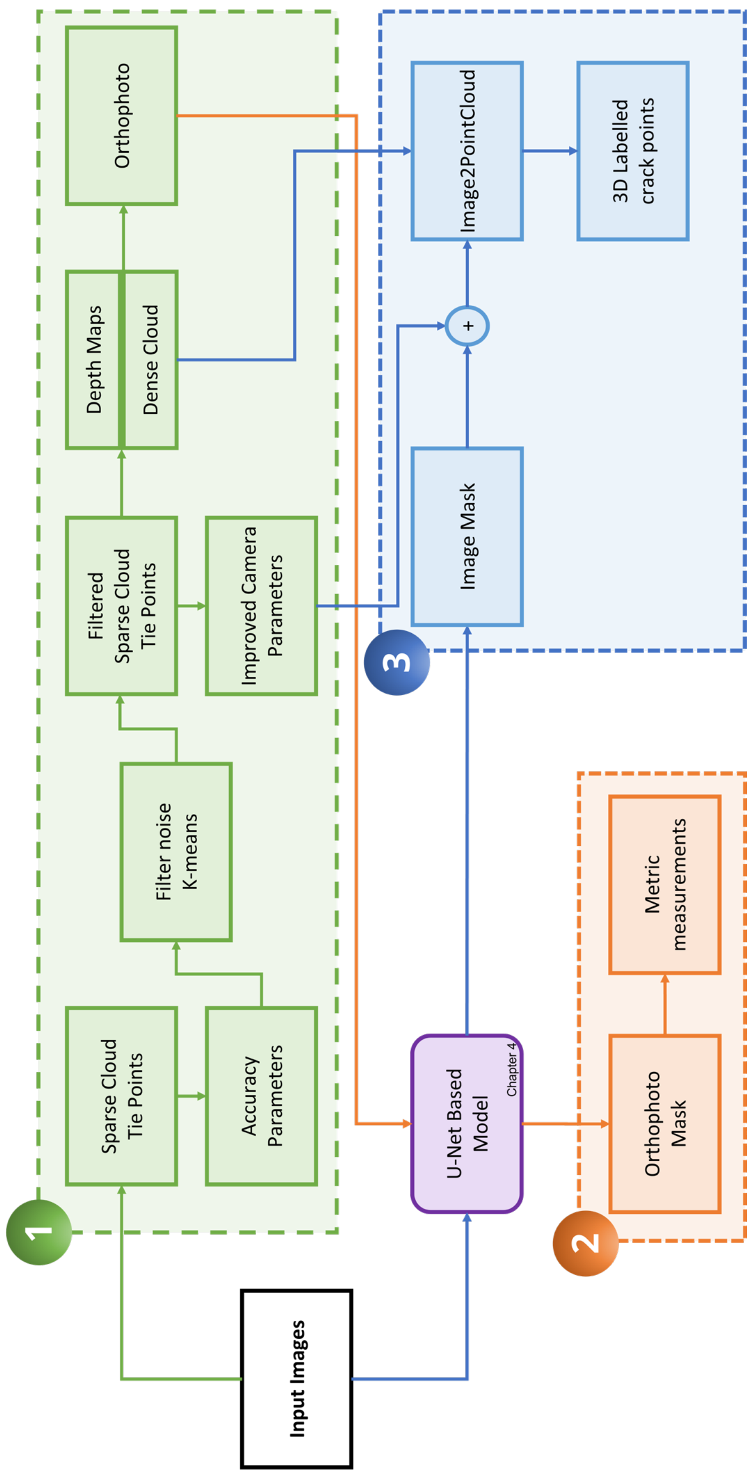


Figure 5.1 Methodology WorkFlow

5.1.1 Improvement of photogrammetric models: K-Means Clustering

First of all, the proposed outlier detection method does not consider a single parameter related to accuracy but simultaneously all calculated parameters by applying a clustering machine learning algorithm, K-Means, to reach a compromise model between the data of points available and the noise reduction associated with the 3D definition. The purpose is to derive a tie point cloud (sparse point cloud) made up of only high-quality tie points and optimize the camera model.

K-Means

Unsupervised learning, also known as unsupervised machine learning [15], analyses and clusters unlabelled information using machine learning techniques. Without the need for human interaction, these algorithms uncover hidden patterns or data groupings. Unsupervised learning models are utilized for three fundamental tasks: clustering, association, and dimensionality reduction. Clustering is the most significant unsupervised learning problem, and it deals with discovering a structure in a collection of unlabelled data.

K-Means is a technique that divides a dataset into a set of groups based on the number of clusters supplied by the user [17]. The program examines the data to identify data points that are organically similar and assigns each point to a cluster of points with similar features. The data can then be labelled into different classes based on the features of each cluster. The algorithm converges when there is no further change in the assignment of instances to clusters. The whole process is implemented using Scikit-learn [108], a free machine learning library for Python.

Finding the optimal number of clusters is an important part of this algorithm. A commonly used method for finding the optimal K value is Elbow Method [109]. The Elbow method (Figure 5.2) is a technique that we use to determine the optimal number of centroids (K) to employ in a K-Means clustering algorithm by calculating the Within-Cluster Sum of Squares (WCSS) for each K value.

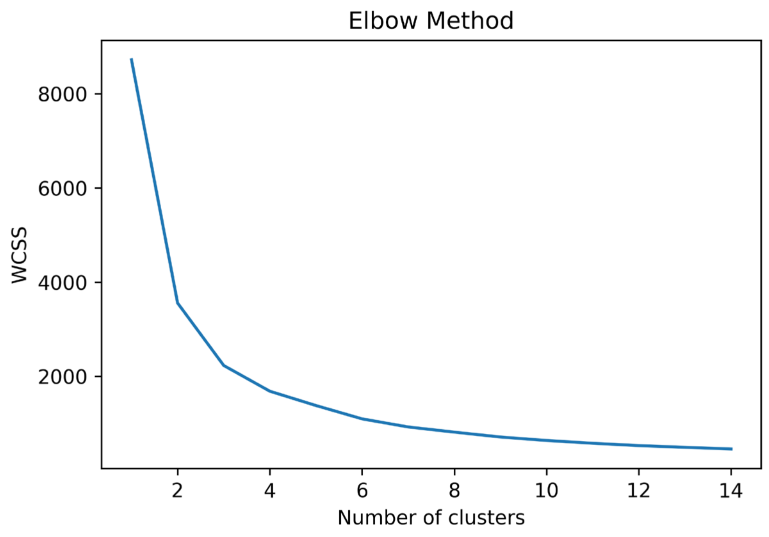


Figure 5.2 Example of Elbow method, clusters = 4.

The sum of squared distances between each point in a cluster and its centroid is WCSS.

$$WCSS = \sum_{C_k}^{C_n} \sum_{d_i in C_i}^{d_m} distance(d_i, C_k)^2 \tag{1}$$

where

C = cluster centroids

d = data point in each cluster

The WCSS value will begin to decrease as the number of clusters increases. When examining the graph, we can see that it shifts rapidly at one point, forming an elbow. At this point, the graph begins to move almost parallel to the X-axis. The optimal value of K, or the optimal number of clusters, corresponds to this point. Thus, in Figure 5.2, the optimal number of clusters for the data is 4.

The goal is to derive a sparse point cloud made up of only high-quality tie points, filtering the sparse cloud by the points belonging to the noise cluster and optimizing the camera model. The choice of cluster membership in the High accuracy or Noise group was determined by evaluating the average angle values. This decision is based on the study of the literature, particularly in the single feature analysis. Previous studies [110,111], have demonstrated that the angle value has the most substantial impact on noise. Clusters with larger average angles were placed in High accuracy, and the group called "Noise" includes the clusters with smaller average angular values. It can be noted that the High accuracy group turns out to be the best-fitting point set (set of points that closely matches the geometrical shape). The tie points of the Noise group are removed from the model, and then the camera model is optimised with the selected default coefficients [f, k1, k2, k3, cx, cy, p1, p2].

- **f:** focal length (in pixels)
- **c_x, c_y:** principal point offset (in pixels)
- **K_1, K_2, K_3:** radial distortion coefficients (dimensionless)
- **P_1, P_2:** tangential distortion coefficients (dimensionless)

The dense cloud from Multi-View Stereo (MVS) can then be generated, as it will use the estimated camera positions generated after the tie point filtering.

Accuracy Parameters

The evaluation of the quality of the photogrammetric design within SfM methods can be done using several features derived both in the acquisition phase (i.e., the number of images contributing to the 3D reconstruction of a tie point or the angle of intersection of homologous rays) and in the image processing phase (i.e., reprojection error) [112]. In this method, the images were processed in Agisoft Metashape [113], and the analysed features are reported below.

Reprojection Error

The reprojection error is a geometric error corresponding to the image distance between a projected point and a measured one [114,115]. It is used to quantify how closely an estimate of a 3D point recreates the point's true projection.

For the computation of the 3D coordinates of a tie point, the parameters of the interior and exterior orientation of the camera and the image coordinates of the point are used. Once its coordinates are computed, the 3D point is reprojected on all the images where it appears. The distance between the image point and the reprojected point on a single image is the reprojection error. This error is also referred to as RMS image residual [114]. Figure 5.3 is an example of how the reprojection error is obtained graphically.

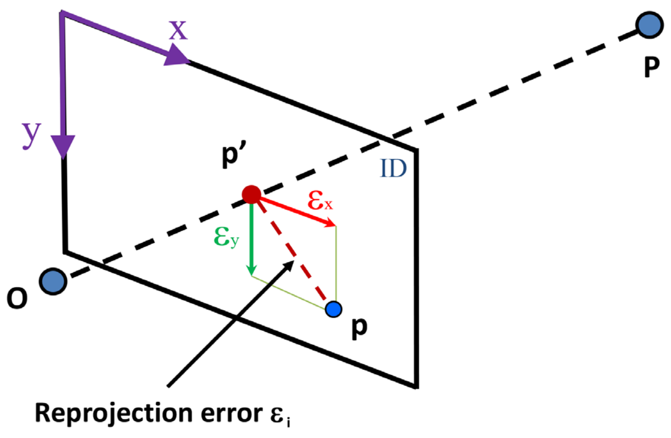


Figure 5.3 Reprojection error

Mathematically, the reprojection error is obtained as follows:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{pmatrix} (R_C^T - R_C^T t_c) \begin{pmatrix} P \\ 1 \end{pmatrix} = KP_c \begin{pmatrix} P \\ 1 \end{pmatrix} \tag{1}$$

$$\varepsilon_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix} - \left[\begin{pmatrix} 0 & 0 & 1 \end{pmatrix} KP_c \begin{pmatrix} p_i \\ 1 \end{pmatrix} \right]^{-1} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} KP_c \begin{pmatrix} p_i \\ 1 \end{pmatrix} \tag{2}$$

where

f_u,f_v = focal lengths in the *u* and *v* directions

u_c, v_c = principal point offset

t_c = position of the camera centre in the object space

R_c = rotation matrix

K = matrix of internal parameters

P = vector projected space coordinate p_i

KP_c = projection matrix

Each tie point extracted is associated with a reprojection error value ε_i, which is the module of the sum of the reprojection errors computed for the number of cameras that see the i-th tie point.

Angle Between Homologous Rays

It is known that one of the parameters that most influence the accuracy of a photogrammetric project is the base/height ratio [116].

In this thesis, by estimating the angle between the two projection lines (called the "intersection angle"), the Base/Height ratio is analysed. The photogrammetry software used does not give the value of this angle in the output, so we implemented an algorithm in the Python environment.

In order to extract and calculate the parameters of interest, two libraries are used: NumPy [117], which is used to add support for large multi-dimensional arrays and matrices, and a large collection of advanced mathematical functions for operations on these arrays; Pandas [118] is a BSD-licensed open source library that provides high-performance, easy-to-use data structures.

Projection Centre (*O*) and tie point (*k*) are the input parameters used to calculate the angle of intersection. Given the k-th tie point seen from two images, *i* and *j*, the direction vectors u_i and v_j are given by the relations:

$$\begin{aligned} u_i &= [E_{Oi} - E_k; N_{Oi} - E_k; h_{Oi} - h_k] \\ v_j &= [E_{Oj} - E_k; N_{Oj} - E_k; h_{Oj} - h_k] \end{aligned} \tag{3}$$

where the subscript O_i indicates the projection centre of the i-th frame and O_j of the j-th frame, *E, N,* and *h* is the elevation.

The relation gives the intersection angle α:

$$\cos\alpha = \frac{u \cdot v}{|u| \cdot |v|} \tag{4}$$

The intersection angle calculation was made using all the image pairs that see the *i-th* tie point, calculating the intersection angle for each pair and finally calculating the average intersection angle between the n frames that see the point (Figure 5.4). Finally, with each tie point extracted, the method associates the average angular value obtained. The whole process is implemented in Python.

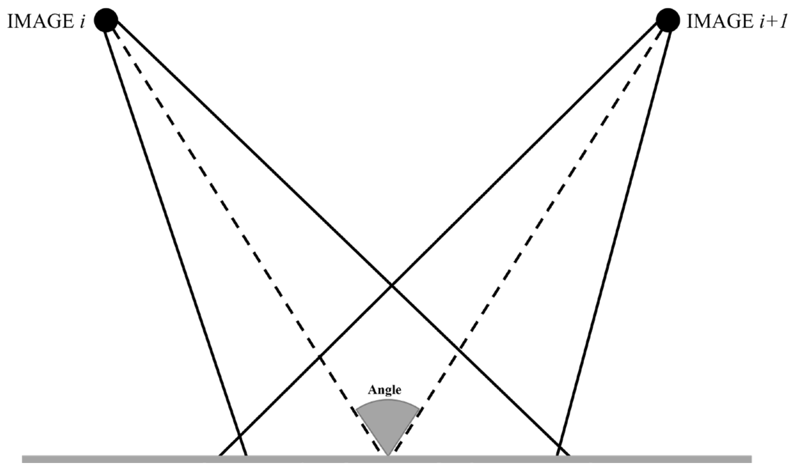


Figure 5.4 Angle Between Homologous Rays

Image redundancy

This parameter is the number of input photogrammetric images within the SfM process for the reconstruction of the i-th TP in 3D space. With the same other parameters of photogrammetric accuracy, it is assumed that as the image redundancy increases, the metric quality of the TP point cloud improves.

$$N_i = \sum_j n_{j_{TPi}} \tag{5}$$

where $n_{j_{TPi}}$= camera for the reconstruction of the i-th TP.

Projection Accuracy

This parameter corresponds to the placement of points in relation to their local neighbours. The accuracy of Tie Point projections is dependent on the scale at which these points were identified. Metashape uses information about scale to weight Tie Point reprojection errors. It is also related to the key point size; key points with lower standard deviation values are located more precisely in the space where they were found [119]. The projection accuracy is computed as follows:

$$\sum_i \frac{s_i}{n} \tag{6}$$

where,

— s_i is the image scale at which corresponding projections are measured on the ith image
— n is the number of images where the tie point is measured.

5.1.2 Crack metric measurements

To quantify the width and area of the cracks, as well as to evaluate the sensitivity of the model in width segmentation, the trained model is applied to an orthophoto generated by the photogrammetric software Metashape with a known pixel size. The output of the model is a binary image, which is an orthophoto mask.

Method for Determining Crack Area and Width Measurements

The crack measurements are calculated using several functions implemented in MATLAB and applied to the model output mask. Specifically, the 'bwmorph' function [120] and the 'bwdist' [121] function are used.

The 'bwmorph' applies a specific morphological operation to the binary crack mask orthophoto. The first operation performed is the so-called 'remove' operation which removes pixels inside the cracks and keeps only the border pixels. The removed pixels are used together with the known pixel size to calculate the crack area to be evaluated. Continuing with the calculation to obtain the crack width, the same function allows us to obtain the skeleton of the cracks using the 'skel' operation. The function reduces binary objects to 1-pixel wide representations. This can be useful for feature extraction and/or representing an object's topology. This operation will be executed repeatedly until the image no longer changes.

Afterwards, the function 'bwdist' computes the Euclidean distance (Formula) between the skeleton and the border pixels. In 2-D, the Euclidean distance between (x_1, y_1) and (x_2, y_2) is:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}$$

The distance thus calculated was used to produce a raster containing the width of the cracks.

5.1.3 From the crack mask to the point cloud: Image2PointCloud

3D crack analysis enables a more detailed and comprehensive visualization of cracks, allowing to observe them from different angles and perspectives. This approach enhances understanding of the spatial relationships between cracks and other structural elements. In addition, it enables the analysis of not only planimetric propagation but also crack propagation in 3D space, thereby facilitating a more robust evaluation. From the perspective of a scan-to-BIM methodology, and therefore of reality-based modelling from a point cloud, the opportunity to have the three-dimensional coordinates of the cracks affecting an infrastructure available ensures both an analysis of its state of health and proper management of the necessary interventions. In this way, the BIM model is enriched with further information on the state of health of the infrastructure, becoming a useful tool for the correct quantification of damage and support for the identification of the costs of the interventions to be carried out.

Image2PointCloud

The Image2PointCloud enables the transformation of object coordinates (3D) to 2D coordinates in the image plane (Figure 5.5). This process has as input the image crack mask, parameters of the camera, and the produced by the photogrammetric software. With these, each 3D point can be projected onto the image space given the exterior, interior orientation and distortion of the respective image obtained after improving the accuracy of 3D photogrammetric models (Subsection 5.1.1)

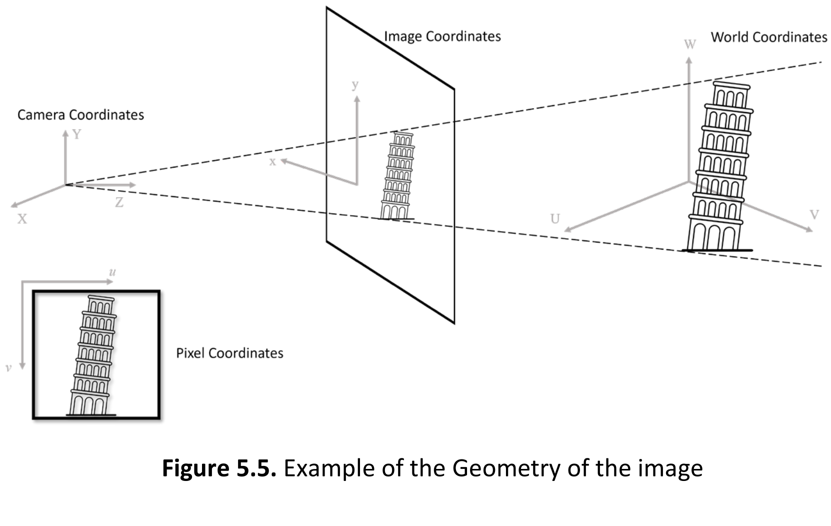


Figure 5.5. Example of the Geometry of the image

The camera projection matrix can be expressed as:

$$s\,p = K[R|t]P_w \tag{1}$$

Putting together the equations for the interior and exterior orientation parameters, we can write:

$$s\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}f_x & 0 & c_x\\0 & f_y & c_y\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_x\\r_{21} & r_{22} & r_{23} & t_y\\r_{31} & r_{32} & r_{33} & t_z\end{bmatrix}\begin{bmatrix}X_w\\Y_w\\Z_w\\1\end{bmatrix} \tag{2}$$

Where,

- s = if an image from the camera is scaled by a factor, all of these parameters should be scaled (multiplied/divided, respectively) by the same factor
- p = image coordinates (*u,v*)
- K = calibration matrix (intrinsic matrix). Composed of the focal lengths f_x and f_y, expressed in pixel units, and the principal point (c_x,c_y), usually near the centre of the image
- R, t = 3D rotation and translation matrix, describing the transfer of coordinates from the external system to the camera system
- P_w = 3D point coordinates (X_w, Y_w, Z_w)

Real lenses usually have some distortion, radial distortion, and tangential distortion. So, the above model is extended as:

$$\begin{bmatrix}u\\v\end{bmatrix} = \begin{bmatrix}f_x x'' + c_x\\f_y y'' + c_y\end{bmatrix}$$

Where,

$$\begin{bmatrix}x''\\y''\end{bmatrix} = \begin{bmatrix}x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x'y' + p_2(r^2 + 2x'^2)\\y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_2 x'y' + p_1(r^2 + 2y'^2)\end{bmatrix} \tag{3}$$

$$r^2 = x'^2 + y'^2 \;\; , \;\; \begin{bmatrix}x'\\y'\end{bmatrix} = \begin{bmatrix}X_c/Z_c\\Y_C/Z_c\end{bmatrix}$$

- X_c, Y_c, Z_c = point coordinates in the local camera coordinate system
- k_1, k_2, k_3 = radial distortion coefficients (dimensionless)
- p_1, p_2 = tangential distortion coefficients (dimensionless)

The algorithm performs the transformation knowing all the parameters of the camera extracted from Metashape. In this manner, the one-to-one relationship between the pixels belonging to the crack (u, v) on the image frame and the points in the 3D reference system of the object will be known. The following algorithm implemented in Python is applied with the help of OpenCV. OpenCV is a Python library aimed at providing the tools needed to solve computer vision problems. It contains a mix of low-level image-processing functions and high-level algorithms [122].

The result, Image2PointCloud, provides the capability to distinguish these points in the dense cloud by adding a label to them.

5.2 Application of the Methods in case studies

This chapter will start by presenting the application of the crack segmentation model proposed in Chapter 4 and the methods described in Section 5.1 through a series of real case studies. These case studies provide an overview of the applicability and effectiveness of the proposed model and methods in crack segmentation and crack evaluation.

Two case studies, serving as the main focus of this investigation, comprise a road pavement and a retaining wall. These environments are chosen due to their distinct structures, conditions, and exposure to various external stress factors, making them ideal candidates to demonstrate the versatility of the proposed model.

5.2.1 Case studies

Road pavement case study

The case study consists of a single-carriageway provincial road (*strada provinciale* in Italy, abbreviation SP) with two lanes, each approximately 3.5 metres wide and one for each direction of traffic (Figure 5.6). The section was chosen to present an articulated crack state; in fact, there are alligator, longitudinal, transverse and edge cracks.

The UAV-based imagery of this study area was acquired using the DJI Matrice 300 RTK UAV and DJI Zenmuse P1 camera, described in the last subsection of this chapter. The automatic flight plan for the acquisition of nadir photogrammetric images was programmed by DJI Pilot to achieve an 80% forward overlap rate and 80% side overlap rate with a 30 m flight height above the relative take-off point. To ensure and maintain flight accuracy, the HxGN SmartNet GNSS reference stations in the Matrice 300 RTK was configured and used to obtain highly accurate location information in both vertical and horizontal directions.

The total number of acquired images is 452, planned with the project requirements in consideration – a Ground Sampling Distance (GSD) of about 0.3 cm – and, simultaneously, with the aim of ensuring a high level of automation in the subsequent steps of data processing. In order to process the photogrammetric data, Metashape Professional Edition by Agisoft (ver. 1.8.4 build 14671) has been used [123]. The following parameters were set to build the sparse cloud consisting of tie points: in the "Align Photos" phase, Accuracy = High (original photos), Key Point limit = 60000, tie point limit = no. The appropriate reference system for the coordinates is selected: RDN2008/UTM zone 33N, identified by code EPSG:7792 and ellipsoidal heights.

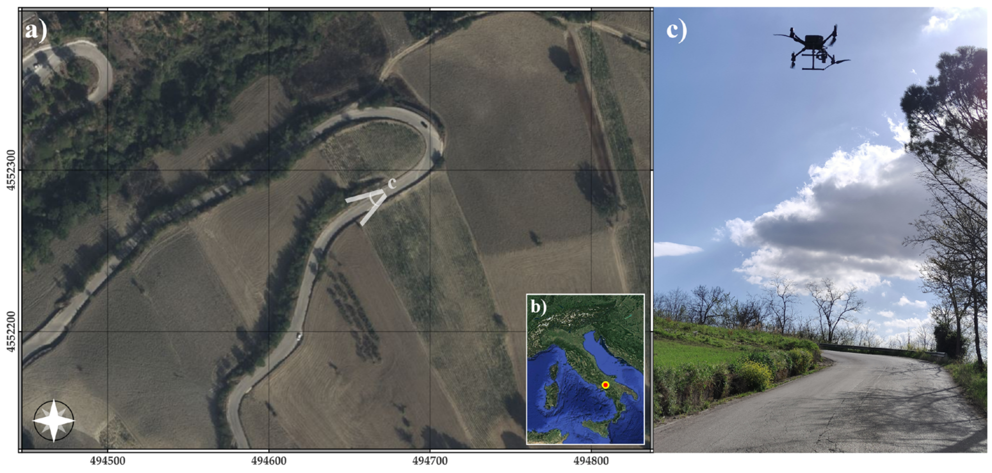


Figure 5.6 Test area: a) location map; b) the red and yellow dot points out the test area (Southern Italy); c) photo taken the day of the survey.

Retaining wall case study

To test the robustness of the method, it was also applied to an elevated structure, specifically a concrete retaining wall adjacent to a building, characterized by a complex crack pattern (Figure 5.7). The texture of the wall is fairly homogeneous but displays areas of deterioration due to dampness and rain.

In this case study, we used the same UAV and camera described in the next subsection of this chapter – the DJI Matrice 300 RTK UAV and the DJI Zenmuse P1 camera. To obtain highly accurate location information, we used the HxGN SmartNet GNSS network. The flight was conducted in manual mode, and the portion used for the analysis consists of 20 images with a Ground Sampling Distance (GSD) of approximately 0.03 cm. The parameters used to process the photogrammetric data in Metashape Professional Edition were identical to those applied in the road pavement case study, Accuracy = High, Key point limit = 60000, tie point limit = no limit and the reference system RDN2008/UTM zone 33N (EPSG:7792 and ellipsoidal heights).

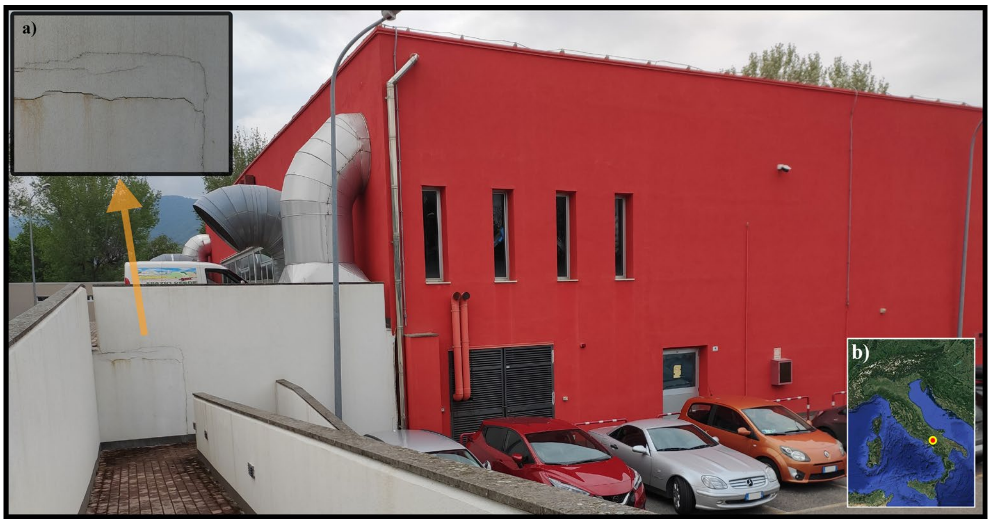


Figure 5.7 Test area: a) area under analysis; (b) the red and yellow dot points out the test area (Southern Italy).

Equipment Utilized for Photogrammetric Surveys in Case Studies

Both surveys were conducted using the DJI Matrice 300 RTK UAV (Figure 5.8) equipped with the DJI Zenmuse P1 camera (Figure 5.9) and a professional quadcopter equipped with a GNSS RTK receiver. Table 2 lists the fundamental characteristics of this UAV in detail.

Figure 5.8 Matrice 300 RTK.

For data acquisition intended for photogrammetric processing, we used the DJI Zenmuse P1 camera equipped with a 35mm lens, as shown in Figure 5.9. Camera characteristics are shown in Table 3.

Figure 5.9 Zenmuse P1 Camera

Table 2. UAV DJI Matrice 300 specs.

DJI Matrice 300 RTK	
Dimensions	810×670×430 mm (L×W×H)
Weight (with single downward gimbal)	Approx. 6.3 kg (with two TB60 batteries)
Max Takeoff Weight	9 kg
RTK Positioning Accuracy	When RTK is enabled and fixed: 1 cm + 1 ppm (Horizontal) 1.5 cm + 1 ppm (Vertical)
Max Flight Time	55 min
GNSS	GPS+GLONASS+BeiDou+Galileo
Max Transmitting Distance	8 km

Table 3. DJI Zenmuse P1 specs.

DJI Zenmuse P1	
Dimensions	198×166×129 mm (L×W×H)
Sensor size (Still)	35.9×24 mm (Full-frame)
Effective Pixels	45MP
Pixel size	4.4 μm
Supported Lenses	DJI DL 24mm F2.8 LS ASPH DJI DL 35mm F2.8 LS ASPH DJI DL 50mm F2.8 LS ASPH
Photo Size	3:2 (8192×5460)
Minimum photo interval	0.7 s
Aperture Range	f/2.8-f/16

5.2.2 U-Net-based Model: Case Studies Results

The results of the U-Net-based crack segmentation model applied to the case studies described above are presented. This was done to test the model at an early stage, without having generated the photogrammetric clouds or orthophotos yet. In this way, we were able to assess the model robustness against data from real case studies in the initial phase. Using the images obtained from the road survey and obtaining the masks after applying the proposed model, some examples are shown in Figure 5.10, which contains cracks with different typologies.

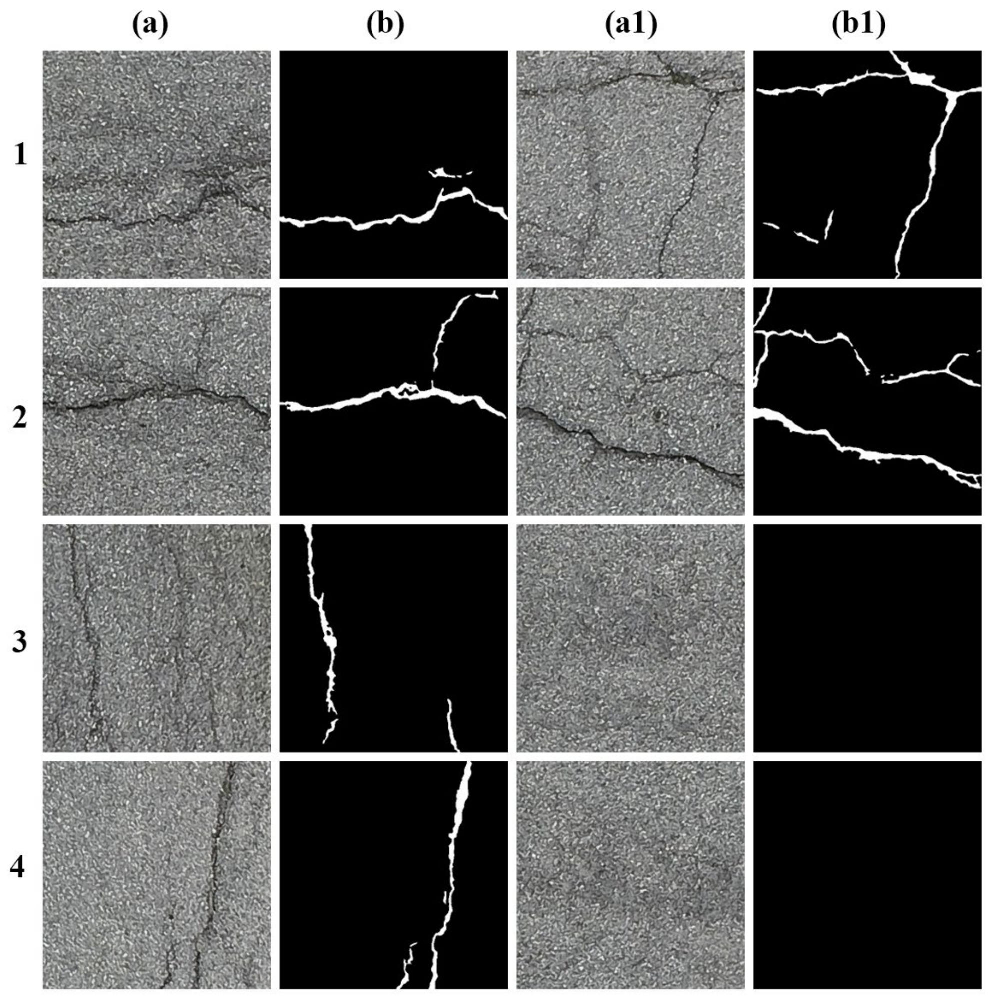


Figure 5.10 Model output in the road pavement case study; (a, a1) Input Road images, (b, b1) Predicted masks by our U-net.

The images given as input to the model, shown in Figure 5.10, columns (a), (a1) belong to the survey data. Columns (b), (b1) show the results obtained on those input images. In almost all cases, the predicted masks appear visually accurate, particularly regarding the crack width (metric assessments of the width are discussed in Subsection 5.2.4). This aspect is crucial, as in the design of maintenance plans, the primary parameter determining the severity levels of various cracks is their width, in addition to their linear development and the area of extension.

Some cracks were not segmented due to the fact that the pixel size must be at least half the width of the crack to be segmented. Thus, with the application of this test case, it can be seen that the parameter affecting segmentation is the image resolution and not the pavement type. Therefore, when designing the survey, it is important to estimate the pixel size as a function of the camera parameters and the acquisition geometry in order to calibrate the photogrammetric survey to segment the cracks with a level of severity that is at least low. We can also highlight the model's ability to adapt to different contexts and surfaces makes it a highly versatile and scalable solution for crack segmentation in various infrastructures.

To test how the model, which was trained with road images (Crack500 dataset), works, it was applied to the case study of the retaining wall using the images obtained by the Zenmuse P1 camera. As a result, the masks of the cracks were obtained. In Figure 5.11, columns (a) and (a1) display the input images provided to the model, which are sourced from the survey data. The results generated based on these input images are presented in columns (b) and (b1).

This model has been trained on an extensive dataset that includes diversity in crack images on roads, which allows it to generalize efficiently and adapt to different contexts and surfaces. Thanks to its robustness and generalization capability, this model obtains results that show accurate segmentation even on surfaces other than road pavements. It can be applied not only to rigid pavements but also to vertical structural elements such as bridges, viaducts, and concrete structures. No tests were performed on masonry structures, as joints between masonry could lead to false positives; in fact, the model is designed for homogeneous textures.

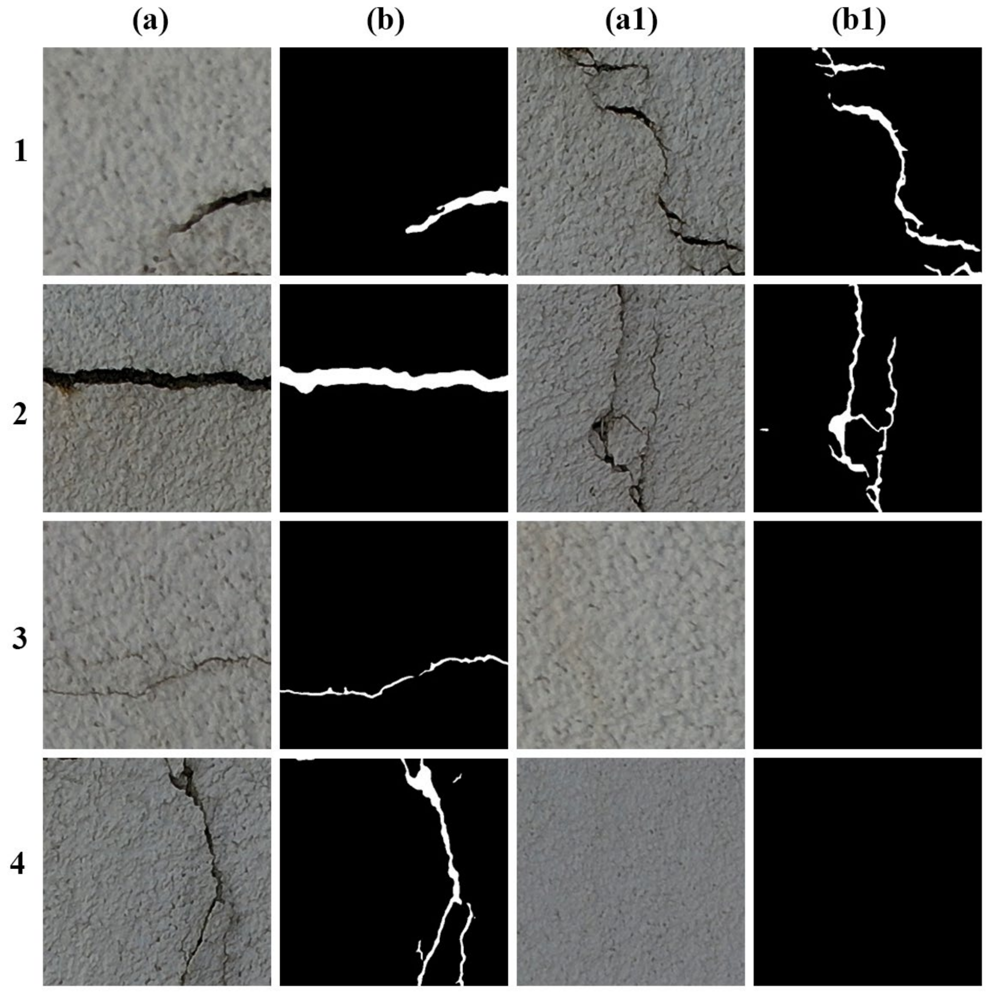


Figure 5.11 Model output in the case study retaining wall; (a, a1) Input wall images, (b, b1) Predicted masks by our U-net.

5.2.3 K-Means Clustering Analysis

The clustering algorithm was chosen to consider not only one accuracy variable but all measurable parameters at the same time. The software Metashape provides the option to extract the desired parameters from the tie points using Python scripting. Metashape estimates and optimizes the camera orientation parameters exclusively based on the tie point information. The accuracy parameters, including reprojection error, projection accuracy, number of pictures, and average angle, were utilized as input data for K-Means analysis.

Analyzing the clusters obtained from K-Means, it was decided to divide them into two groups: the High accuracy group and the Noise group. This division was based on checking the average angle values because it is known from the literature that the angle value is the one that most affects noise [110,111]. Clusters with larger angles were placed in High accuracy, averaged over the number of tie points belonging to the cluster, and the group called “Noise”, which includes the clusters with smaller average angular values. With the Scikit-learn library, all the data were clustered by the K-Means clustering algorithm. Using 10 as the number of times that the K-Means algorithm will be run with different centroid seeds and 300 as the maximum number of iterations of the K-Means algorithm for a single run.

For the road case study, the number of clusters obtained after analysis of the elbow graph is 4. Figure 5.12 shows, with the black line, the number of clusters chosen for the analysis. Furthermore, by analysing Figure 5.13 belonging to the wall case study, it can be seen that the number of clusters equals 3.

The average values of the various analysed features belonging to the High accuracy and Noise groups are shown in Table 4 for the road pavement case study and Table 5 for the retaining wall case study. The analysis of the various features by K-Means made it possible to consider - in the process of filtering the cloud - all the parameters analysed individually. In particular, Table 4 and Table 5 show the mean values and standard deviations obtained in the clustering process categorised as High accuracy and Noise class.

For the road pavement case study (Subsection 5.2.1), a standard section 1 meter wide was identified for the purposes of the analysis, visible in Figure 5.14 (S-S') in red. For the Retaining Wall case study (Subsection 5.2.1), a section 6 cm wide was identified, visible in Figure 5.15 (S-S') in red. Panel b of Figure 5.14 and Figure 5.15 display the section of the point cloud before the filter was applied; as can be seen in both cases, the sections have much noise, and the geometry is not well-defined. Panel c of Figure 5.14 and Figure 5.15, on the other hand, shows the sections of the point clouds filtered using the proposed K-Means method; the clouds are much less noisy, and the geometry is significantly better defined. This improvement also occurs in other parts of the point cloud.

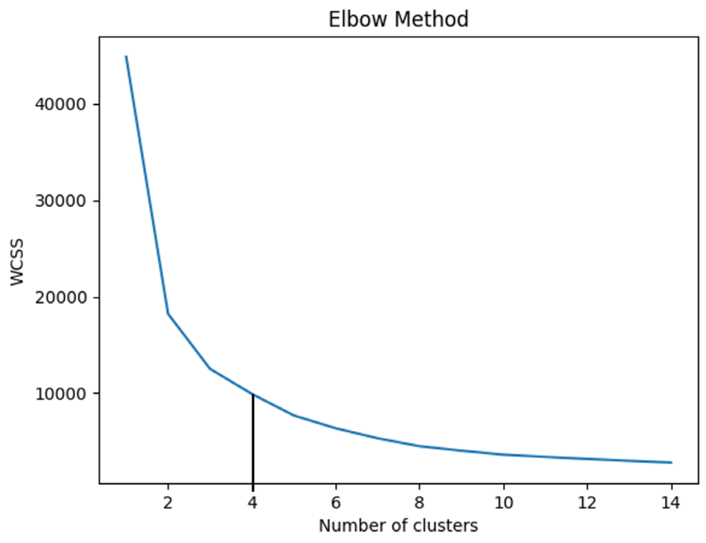


Figure 5.12 Elbow Method for Road pavement data.

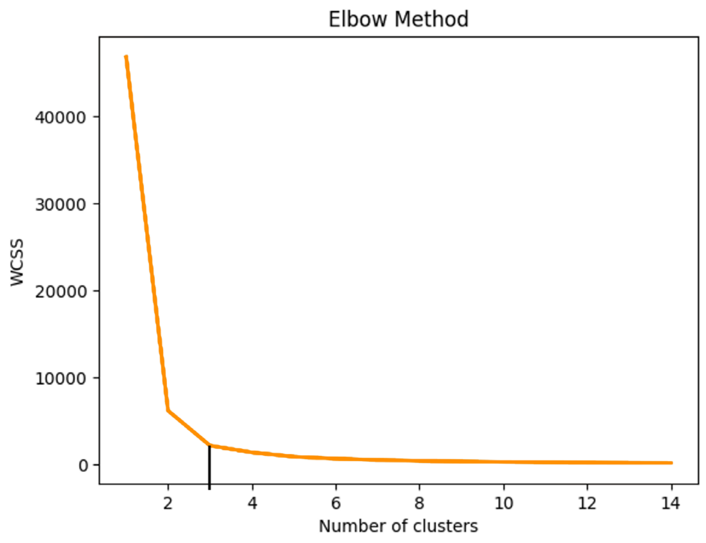


Figure 5.13 Elbow Method for retaining wall data.

Table 4. Road K-Means clustering: feature parameter analysis.

		E_Proj	Acc_Pr	N_Im	Angle °	
K-Means Analysis Road Pavement	Mean	0.77	6.11	5	23.61	High accuracy
	std	0.63	64.55	3	11.40	
	min_value	0.01	0.00	2	1.24	
	25% perc	0.48	0.88	3	15.20	
	50% perc	0.67	2.01	4	22.41	
	75% perc	0.82	4.21	6	29.64	
	max_value	28.09	9468.70	26	105.33	
	Mean	0.50	0.69	3	2.22	Noise
	std	0.64	4.60	1	1.43	
	min_value	0.00	0.00	2	0.05	
	25% perc	0.17	0.03	2	0.96	
	50% perc	0.37	0.14	2	1.99	
	75% perc	0.66	0.61	3	3.39	
	max_value	55.36	825.41	21	5.19	

Table 5. Wall K-Means clustering: feature parameter analysis.

		E_Proj	Acc_Pr	N_Im	Angle °	
K-Means Analysis Retaining Wall	Mean	0.48	6.08	9.82	19.69	High accuracy
	std	0.62	43.91	2.94	3.06	
	min_value	0.06	0.03	6.00	8.83	
	25% perc	0.27	0.64	8.00	17.35	
	50% perc	0.34	1.12	9.00	19.39	
	75% perc	0.47	2.20	11.00	21.83	
	max_value	19.04	3800.12	37.00	33.17	
	Mean	0.19	0.22	2.19	6.55	Noise
	std	0.24	2.46	0.45	1.87	
	min_value	0.00	0.00	2.00	1.44	
	25% perc	0.07	0.01	2.00	5.37	
	50% perc	0.14	0.04	2.00	6.81	
	75% perc	0.25	0.15	2.00	7.88	
	max_value	17.55	615.72	8.00	10.22	

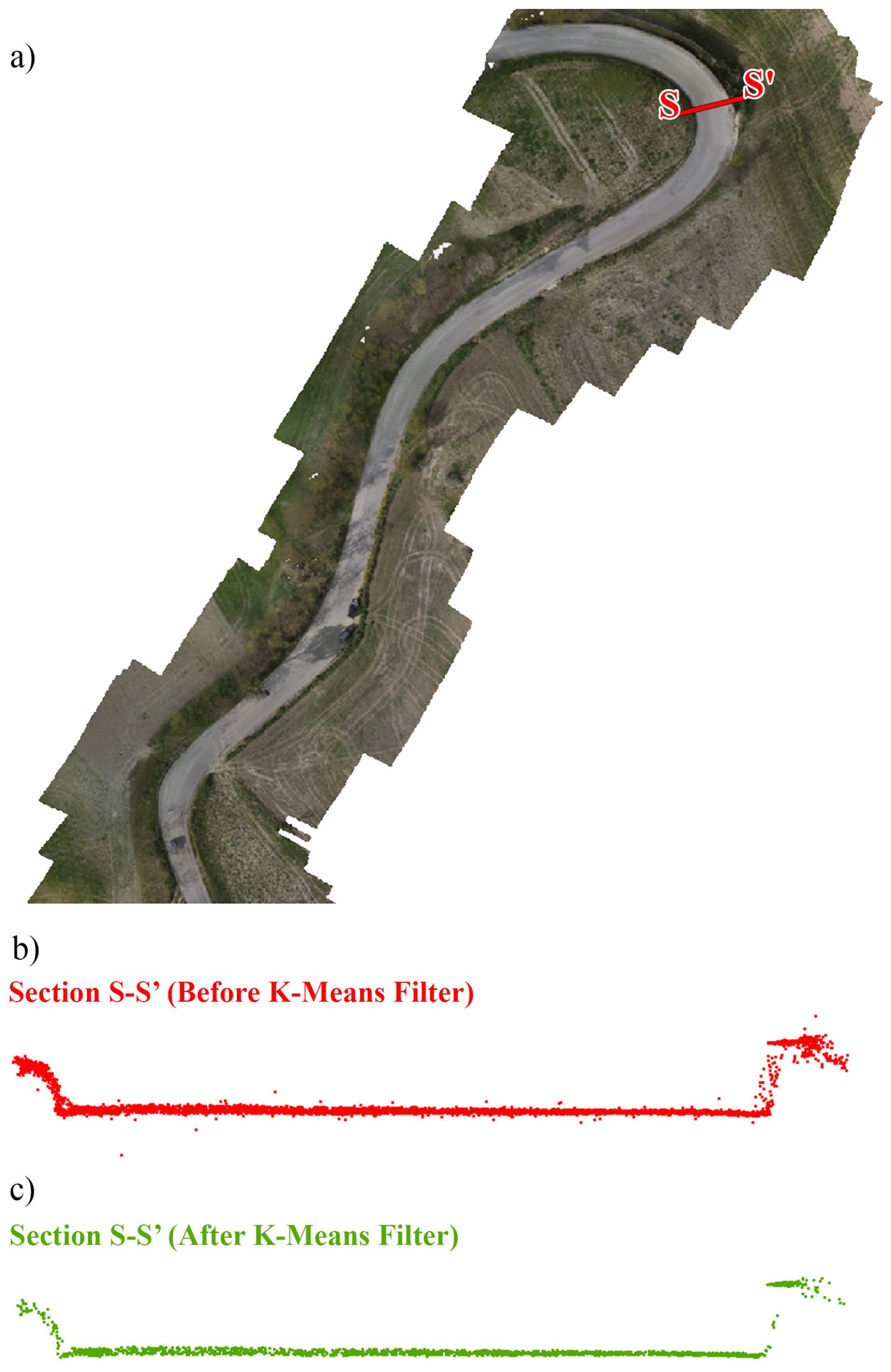

a)
S
S'
b)
Section S-S' (Before K-Means Filter)
c)
Section S-S' (After K-Means Filter)

Figure 5.14 Road pavement section. a) Section S-S' (in red); b) Tie point cloud before the filter; c) Tie point cloud after the filter.

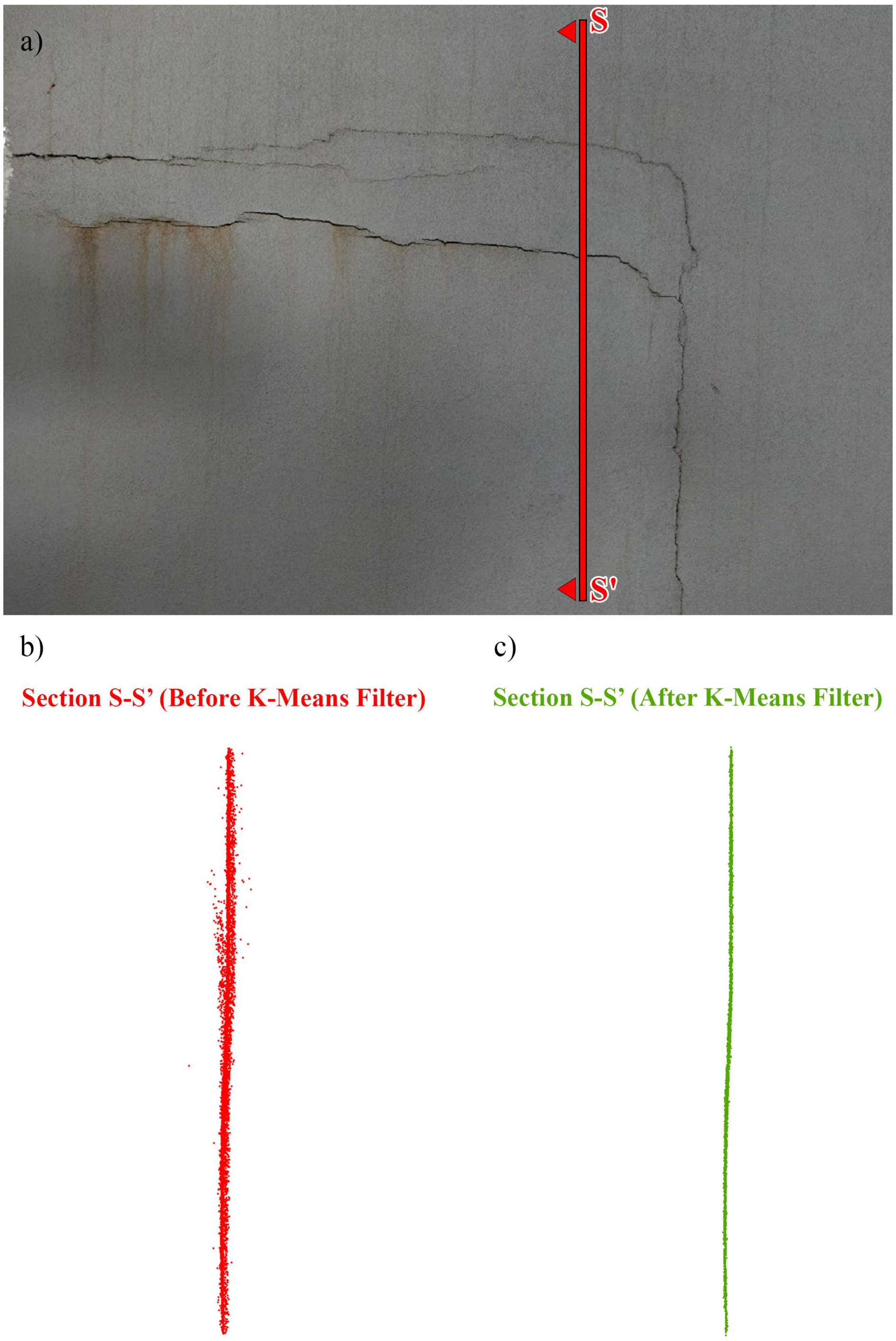

a)
S
S'
b)
Section S-S' (Before K-Means Filter)
c)
Section S-S' (After K-Means Filter)

Figure 5.15 Retaining wall section. a) Section S-S' (in red); b) Tie point cloud before the filter; c) Tie point cloud after the filter.

The filtering process was essential; it enabled the visualization of the displacement (deformation) in the wall that caused the segmented cracks, as shown in Figure 5.15. Particularly noteworthy is the integration of the crack state with the deformation state identified in the 3D model. This integration enables the reconstruction of the ongoing deformation and the potential propagation of the crack state.

The camera model is optimised using the filtered tie point cloud through the software that calculates the coefficients [f, k_1, k_2, k_3, c_x, c_y, p_1, p_2]. Following the tie point filtering, the estimated camera positions can be used to generate dense clouds. Generating a dense point cloud relies on depth maps derived from dense stereo matching. These maps are computed for overlapping image pairs, taking into account their respective exterior and interior orientation parameters, which are estimated through bundle adjustment.

Each camera's multiple pairwise depth maps are combined, using the excess information in overlapping areas to remove inaccurate depth measurements. For each camera, the combined depth maps are converted into partial dense point clouds, which are then merged into a final dense point cloud. An additional noise filtering step is applied in the overlapping areas. The normals within the partial dense point clouds are determined using plane fitting in the pixel neighbourhood of the combined depth maps, while colours are obtained from the images.

Metashape often generates highly dense point clouds, with a density comparable to or even greater than those from LiDAR. These dense point clouds can be modified within the Metashape interface and serve as a basis for further processing steps, including Digital Elevation Model (DEM) creation and subsequent orthophoto generation.

5.2.4 Metric measurements verification

Examining the first case study of the road (Subsection 5.2.1), which analyzes a short section of a provincial road with one carriageway and two lanes in each direction (Figure 5.16). The crack pattern visible in the figure primarily affects one lane.

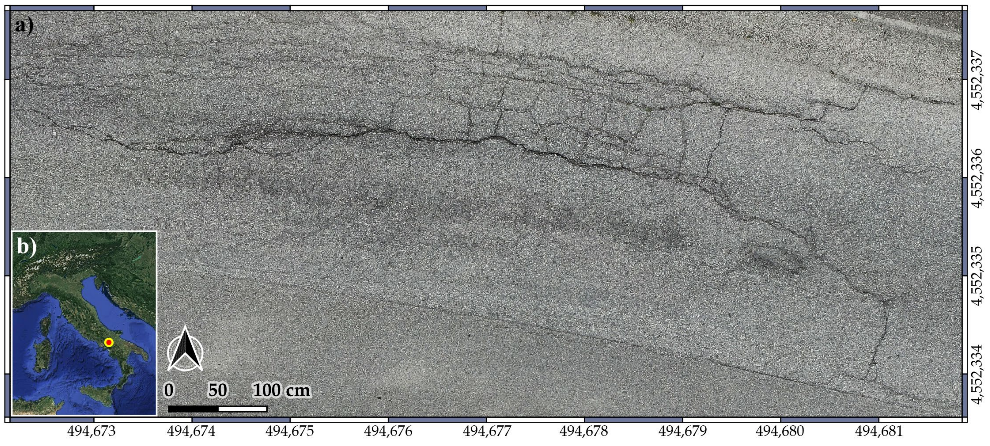


Figure 5.16 a) Orthorectified image of a distressed road pavement used to test our model. Reference system: UTM33/RDN2008; (b) Map of Italy, the red dot indicates the test site.

To quantify the width of the cracks and to assess the sensitivity of the model on width segmentation, the trained model was applied to the orthophoto with a pixel size of 3 mm.

Figure 5.17 shows the results of using the proposed model on the section of road pavement characterized by different types of cracking (mainly Block, Longitudinal and Fatigue Cracking) as well as a surface course with the wear layer being of a texture composition different from that of the Crack500 Dataset. The three boxes 1), 2) and 3) show cracks of different severity levels, corresponding to High, Moderate and Low, respectively. Again, in panels (a) are shown the predicted masks obtained in (b), the masks predicted and classified according to crack width in (c), and the measurements made with the calliper.

The colours in Figure 5.17a are assigned according to the severity levels of the cracks as reported by the Distress Identification Manual of the Federal Highway Administration Research and Technology [124]. The examples highlight the good sensitivity of the model in estimating crack width, a key aspect for classification in terms of severity levels.

Looking at Figure 5.17a, one can notice that our model does not segment cracks with severity levels lower than the low level, mainly because the resolution of the orthophoto does not allow the detection of cracks with amplitude less than or slightly greater than the pixel size. Some cracks characterized by a low severity level were not segmented because the colour difference between the crack and the pavement was not sharp enough, but this is plausible given that the images were taken at a flight height of about 30 m. Cracks with medium/high levels of severity were all segmented, as also observed by traditional surveys carried out in situ. The sample area is approximately 10 meters long and 3.5 meters wide; the methodology employed to estimate the area yields a total area of 34.5m^2 and a crack area of 0.26m^2 for the region analyzed with the orthophoto.

In particular, a high severity level crack is shown in panel 1c, congruent with the crack width estimated from the mask (panels 1a-1b). In panel 2c, a Moderate severity level crack is shown; again, the implemented model accurately segmented the crack as the severity level inferred from the mask (panels 2a - 2b) is congruent with that measured in situ. Finally, panel 3c shows a case halfway between Low and Moderate severity, the crack width being just over 6 mm. In panel 3b, the part considered falls in the Medium severity level (> 6 mm) at the node of the crack junction; as one moves away from the junction (toward the right), the severity level turns Low (< 6 mm). Again, the model was able to segment the crack width with sufficient accuracy; the severity level is congruent with that measured in situ.

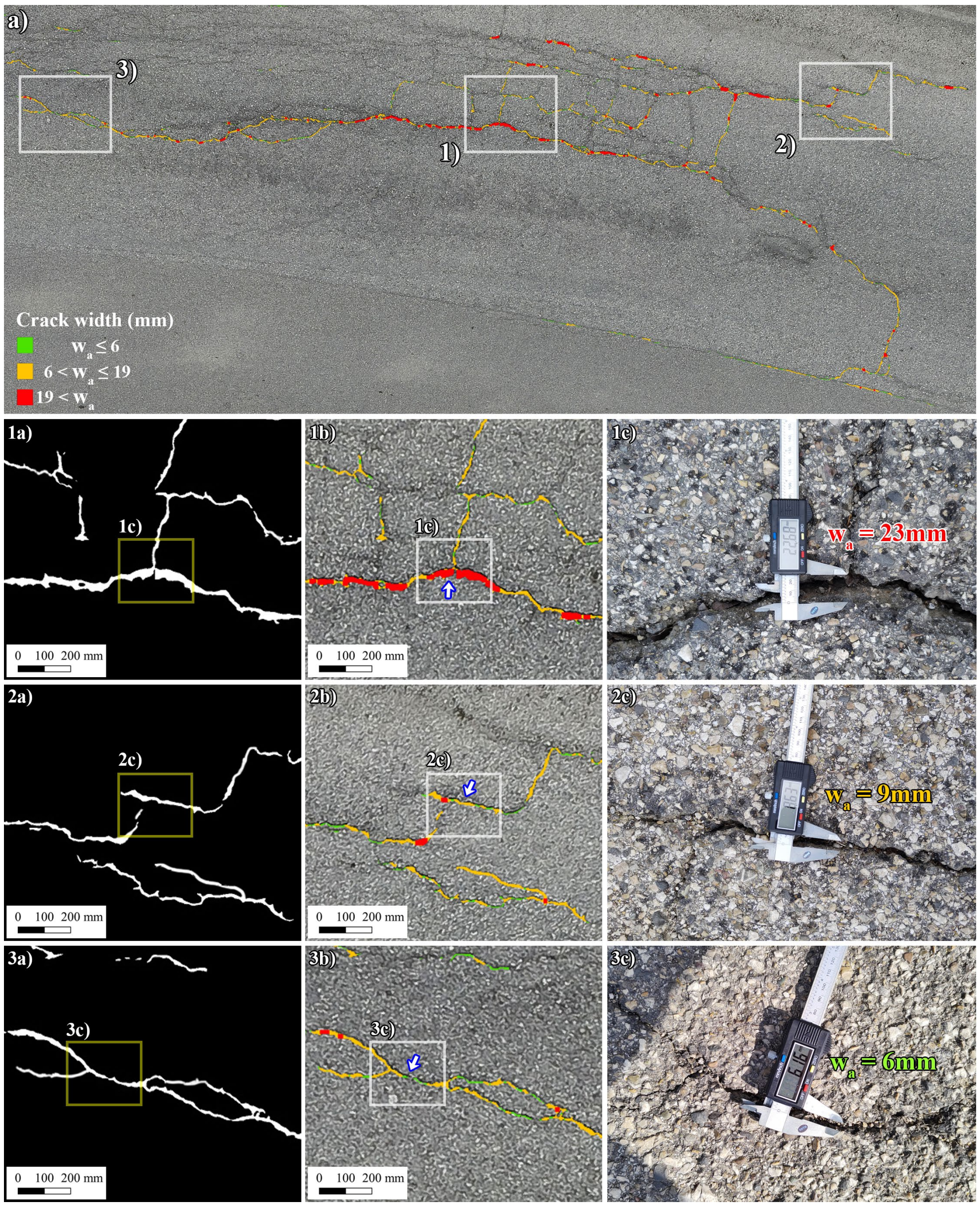


Figure 5.17 Results of a test done on an orthorectified image of a distressed road pavement - pixels size 3 mm; a) Segmented cracks superimposed on the orthophoto; 1a-2a-3a) Excerpt of the output mask; 1b-2b-3b) Excerpt of the orthophoto with overlaid the raster containing the crack width, blue arrow points to the position of the calliper; 1c-2c-3c) In situ measurements with a calliper.

The second case study focuses on a retaining wall (Subsection 5.2.1) and analyzes a cracked section of the wall (Figure 5.18).

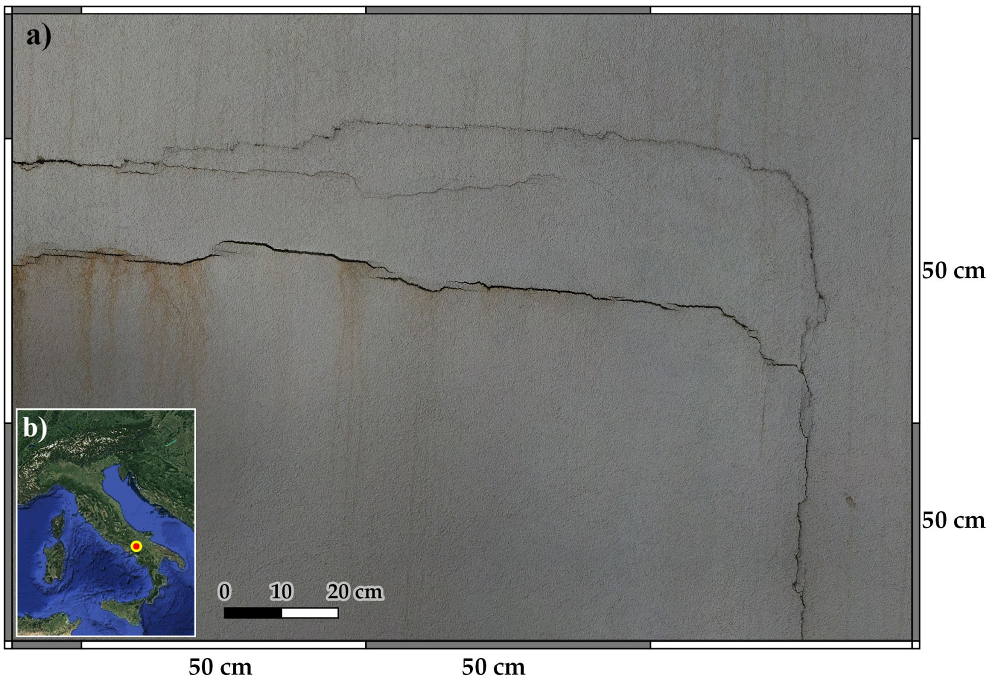


Figure 5.18 a) Orthorectified image of a distressed retaining wall used to test our model; (b) Map of Italy, the red dot indicates the test site.

The trained model was used on an orthophoto with a pixel size of 0.27 mm. Figure 5.19 demonstrates the outcomes of employing the proposed model on a wall section featuring various crack types, surface textures, and colours that differ from those found in the Crack500 dataset.

Boxes 1), 2), and 3) highlight cracks with varying severity levels, corresponding to High, Moderate, and Low, respectively. Panels (a) showcase the predicted masks obtained in (b), while (c) presents the masks predicted and categorized based on crack width, along with the calliper measurements.

The colors in Figure 5.19a are assigned as follows: low severity cracks are less than or equal to 1mm, moderate severity cracks are between 1mm and 2.5mm, and high severity cracks are greater than 2.5mm. The examples emphasize the model's strong ability to estimate crack width accurately, which is crucial for categorizing cracks according to their severity levels.

It is clear that our model does not segment cracks with severity levels below the low level can be observed in Figure 5.19a. The resolution of the orthophoto, which limits the identification of cracks with an amplitude less or just slightly bigger than the pixel size, is the main cause of this. Some low-severity cracks could not be segmented because there was not sufficient colour contrast between the surface and the crack. The segmentation of cracks with medium or high severity levels, however, was comparable with the findings of traditional on-site assessments. The sample area is approximately 1.3 meters long and 0.7 meters wide; the methodology employed to estimate the area yields a total area of 0.9m^2 and a crack area of 0.005m^2 for the region analyzed with the orthophoto.

In particular, panel 1c shows a crack with a high severity level that is compatible with the crack's predicted width according to the mask (panels 1a–1b). A moderate severity level crack is shown in panel 2c. Once more, the model segmented the crack properly, and the severity level inferred from the mask in panels 2a and 2b matched the in situ measurement. Finally, panel 3c shows a case with a crack width of just over 0.5 mm that is almost at the threshold between low and moderate severity. Again, the model applied to a non-road surface proved to be effective and generalised well. It accurately segmented the crack width, and the degree of severity matched the in-situ measurements.

In summary, the performance of segmentation is closely related to the resolution, quality, and to some extent, the exposure of the image. To segment cracks with low or lesser severity levels, it is advisable to use images taken very close to the pavement, preferably from a mobile system mounted on a car and using a medium/high-quality camera capable of achieving a pixel size of at least half the width of the crack. The key point is that cracks with medium/high severity levels are identified and segmented in almost all cases, which is a crucial aspect in decision-making and drafting management plans.

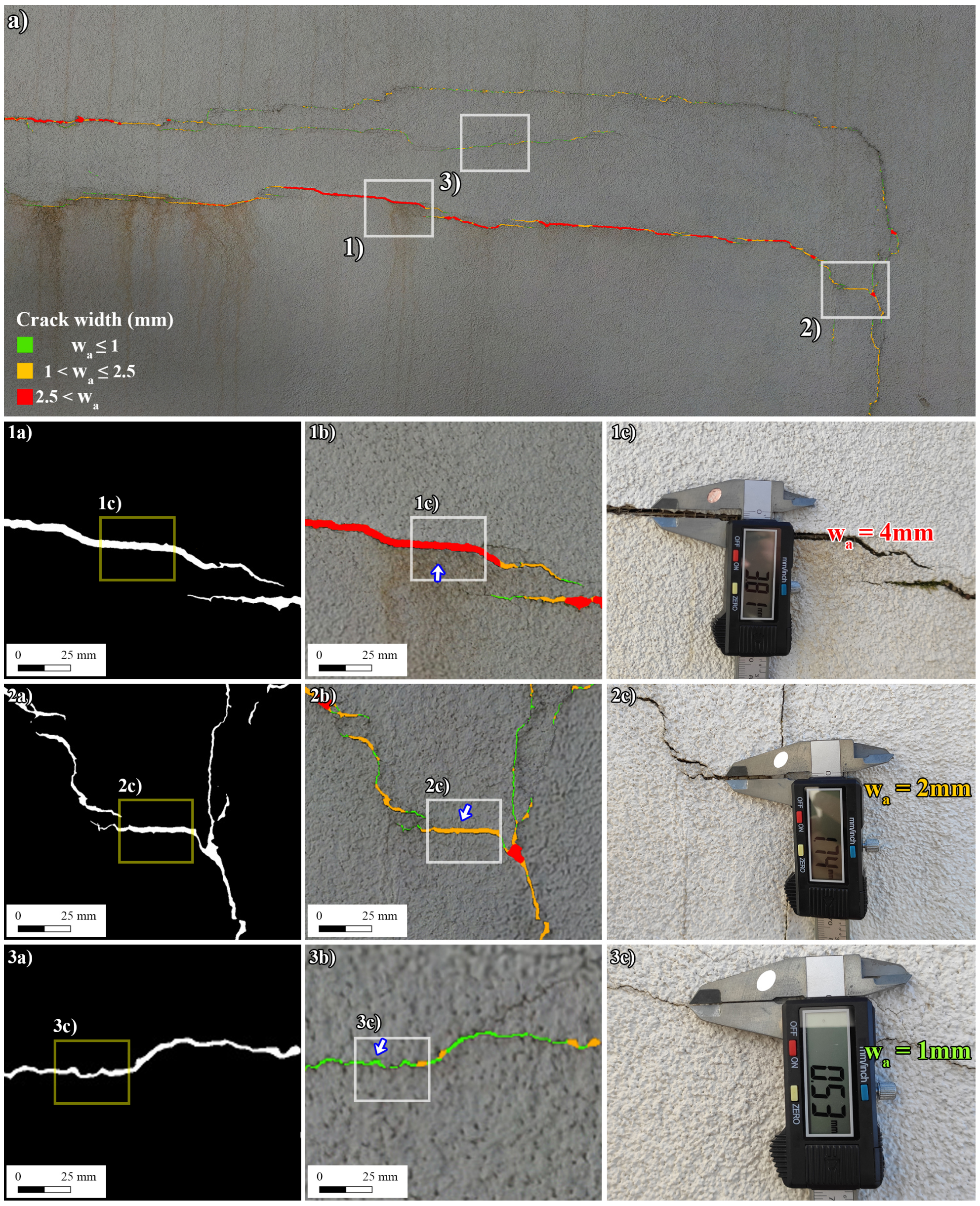


Figure 5.19 Results of a test done on an orthorectified image of a crack retaining wall - pixels size 0.27mm; a) Segmented cracks superimposed on the orthophoto; 1a-2a-3a) Excerpt of the output mask; 1b-2b-3b) Excerpt of the orthophoto with overlaid the raster containing the crack width, blue arrow points to the position of the calliper; 1c-2c-3c) In situ measurements with a calliper.

5.2.5 Image2PointCloud application

To evaluate the performance of the Image2PointCloud method, an image taken by the DJI Zenmuse P1 containing the cracks analysed in Subsection 5.2.4 was used. In the case of the road pavement data (Subsection 5.2.1), the input image used covers a portion of the road. Additionally, are also used the camera parameters exported from Metashape (Table 6) and the respective dense cloud from MVS based on the calculated interior and exterior camera parameters previously mentioned. Figure 5.20 displays the road pavement data results using the 3D point cloud processing software, CloudCompare.

Table 6. Camera parameters obtained from Metashape for the Road pavement case study.

Camera Parameters – Road Image

Focal lengths (f_x, f_y)	$(8245.59, 8245.59)$
Principal point (c_x,c_y)	$(4063.41, 2767.26)$
3D rotation matrix	$\begin{bmatrix} 0.85 & -0.52 & 0.002 \\ 0.52 & 0.85 & -0.013 \\ 0.005 & 0.013 & 0.99 \end{bmatrix}$
Translation Matrix	$\begin{bmatrix} 1957377.14 \\ -4139223.83 \\ -62882.27 \end{bmatrix}$
3D point coordinates (Dense Cloud)	X_r Y_r Z_r
Image coordinates	u_r v_r

In detail, the values X_r Y_r Z_r in Table 6 are the points corresponding to the dense cloud produced. The values to be obtained, u_r and v_r, are the image pixel positions to which the analyzed point of the cloud corresponds.

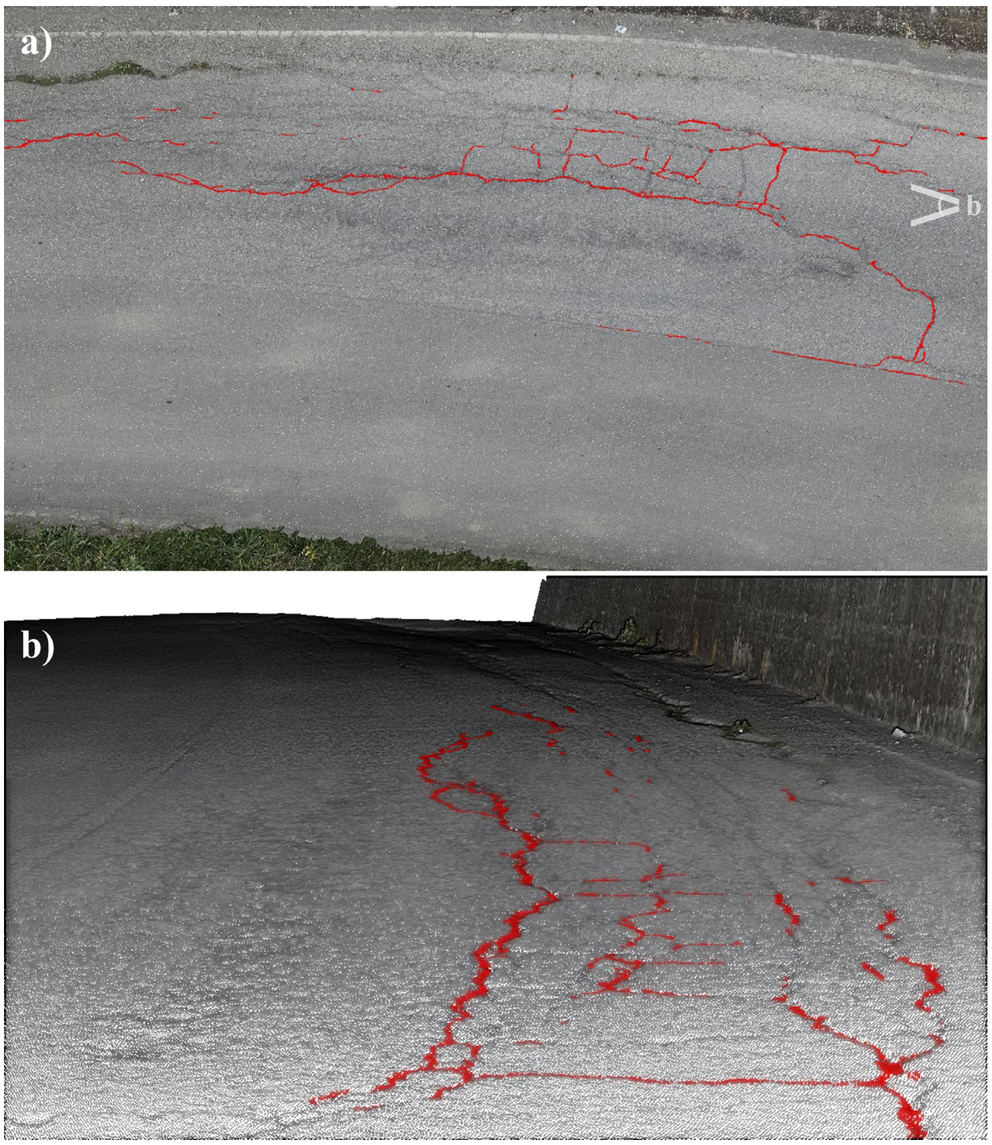


Figure 5.20 a) Dense point cloud of the road, with crack points labelled in red, obtained from the image covering a portion of the road. At the top right of the image is the viewpoint of image b. b) Perspective of the dense point cloud for improved interpretation.

From Figure 5.20, it is possible to observe that the cracks are well associated with the point cloud. As noted in the severity analysis (Subsection 5.2.4), some cracks, characterized by a low severity level, were not segmented due to the insufficient colour difference between the crack and the pavement. However, this is plausible, given that the images were taken at a flight height of about 30 meters.

As parameters, the dense cloud of the road we worked with has a density of about 25000 points per square meter and an average distance between points of approximately 6 millimetres.

On the other hand, for the retaining wall data (Subsection 5.2.1), we used an image which covers a portion of the façade. We also utilized the camera parameters, as shown in Table 7, exported from Metashape, along with the respective dense cloud created. Figure 5.21 displays the results using CloudCompare for better visualization of the crack labels.

Table 7. Camera parameters obtained from Metashape for the retaining wall case study.

Camera Parameters – Wall Image

Focal lengths (f_x, f_y)	$(8245.59, 8245.59)$
Principal point (c_x,c_y)	$(4037.03, 2773.01)$
3D rotation matrix	$\begin{bmatrix} -0.99 & -0.11 & 0.01 \\ 0.02 & -0.08 & 0.99 \\ -0.11 & 0.98 & 0.09 \end{bmatrix}$
Translation Matrix	$\begin{bmatrix} 987192.87 \\ 383786.71 \\ -4413875.28 \end{bmatrix}$
3D point coordinates (Dense Cloud)	X_w Y_w Z_w
Image coordinates	u_w v_w

In Table 7, the values X_w Y_w Z_w represent the points corresponding to the dense cloud. The values to be determined, u_w and v_w, indicate the image pixel positions associated with the analyzed point in the cloud.

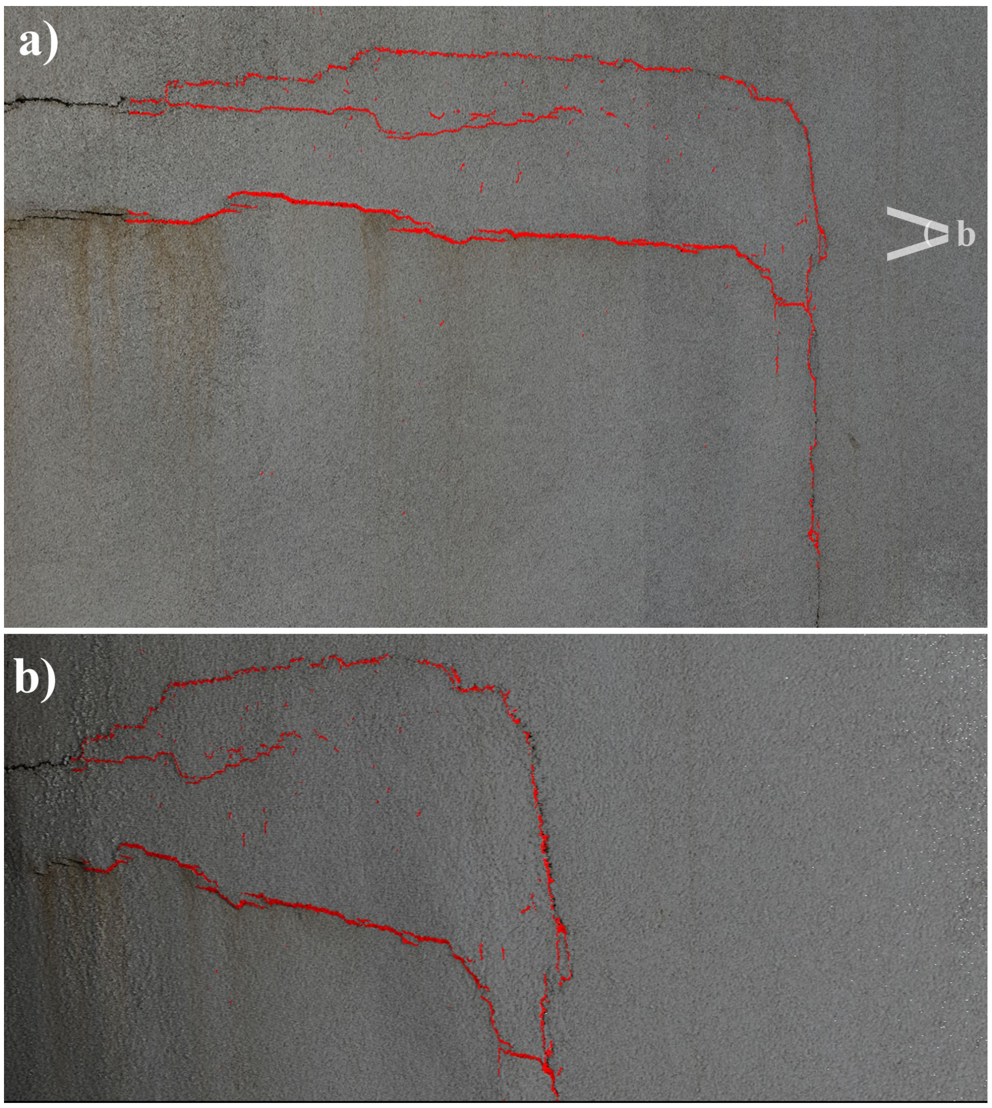


Figure 5.21 a) Dense point cloud of the wall, with crack points labelled in red, obtained from the image covering a portion of the façade. At the top right of the image is the viewpoint of image b. b) Perspective of the dense point cloud for improved interpretation.

Each point in the crack point cloud can be associated with the amplitude value (Subsection 5.2.4) for that section. The data can be imported into the BIM environment, allowing for amplitude information to be available in addition to the 3D geometry.

Conclusions and Outlook

In this thesis, the results of an innovative crack segmentation model based on deep learning were presented. The proposed network structure is a U-Net with a ResNet50 encoder pre-trained with the ImageNet dataset. The encoder component has proven effective in extracting crack features from images, even when they have irregular shapes and intricate textures, thanks to the model's ability to accurately represent global context information. The model was trained with a diverse and comprehensive dataset, to which data augmentation was applied to make the dataset even larger and subsequently validated using typical image segmentation metrics.

Even though the approach followed in this study performed well, there is still much work to be done before pavement cracks can be automatically detected. A drawback of our proposed approach is that it requires a large number of manually drawn pixel-level crack images to build effective and accurate models. This is generally a well-known issue in the literature and is true for almost all ML approaches. The model's performance is directly related to the dataset, and the manual annotation process is time-consuming and subjective. Collecting and labelling data samples takes time and must be done accurately; synthetic data can be introduced in the model learning process.

The validation of the model's performance is generally done on images acquired in nearly optimal conditions, with no noise, obstacles, shadows, or overexposed areas; thus, applying the model to lower-quality images or images with noise could lead to inaccurate results.

For these reasons, the proposed methodology, particularly the U-net-based model integrated with the K-Means clustering method in the cloud filtering process to generate accurate photogrammetric data, was applied to two case studies under real-world conditions. The approach has led to quite promising results, as the severity levels obtained from crack calculations on the resulting masks are consistent with those calculated from traditional in-situ surveys. The improvements made in our model also affect the segmentation of crack width; this aspect is relevant because the width is the key parameter for estimating severity levels that mainly affect the identification of stretches having a priority for interventions.

The ability to adapt to different contexts and surfaces makes this model a highly versatile and scalable solution for the detection and segmentation of cracks in various infrastructures and structures.

A proposal to also use the point cloud of cracks in BIM is the possibility of visualizing only the lesions on a three-dimensional model as a point cloud. Through a Dynamo script, it is possible to transform each crack into a Filled Region to obtain a quantification of the area affected by each lesion. It is also possible to create visual filters to discretize each lesion according to its severity in order to guide the proper planning of the interventions to be made. In this way, the BIM model is enriched with further information on the state of health of the structure, becoming a valid tool for the correct quantification of damage and support for the identification of the costs of the interventions to be carried out.

Going forward, the intention is to optimize the proposed model and test its performance on another dataset type. A more sophisticated crack dataset, including cracks in buildings or bridges, is expected to be built to improve the crack segmentation algorithms.

[Page reserved for Acknowledgements]

Appendix

Table 8. Summary of the Proposed Model: U-Net-based Architecture with a ResNet50 Encoder

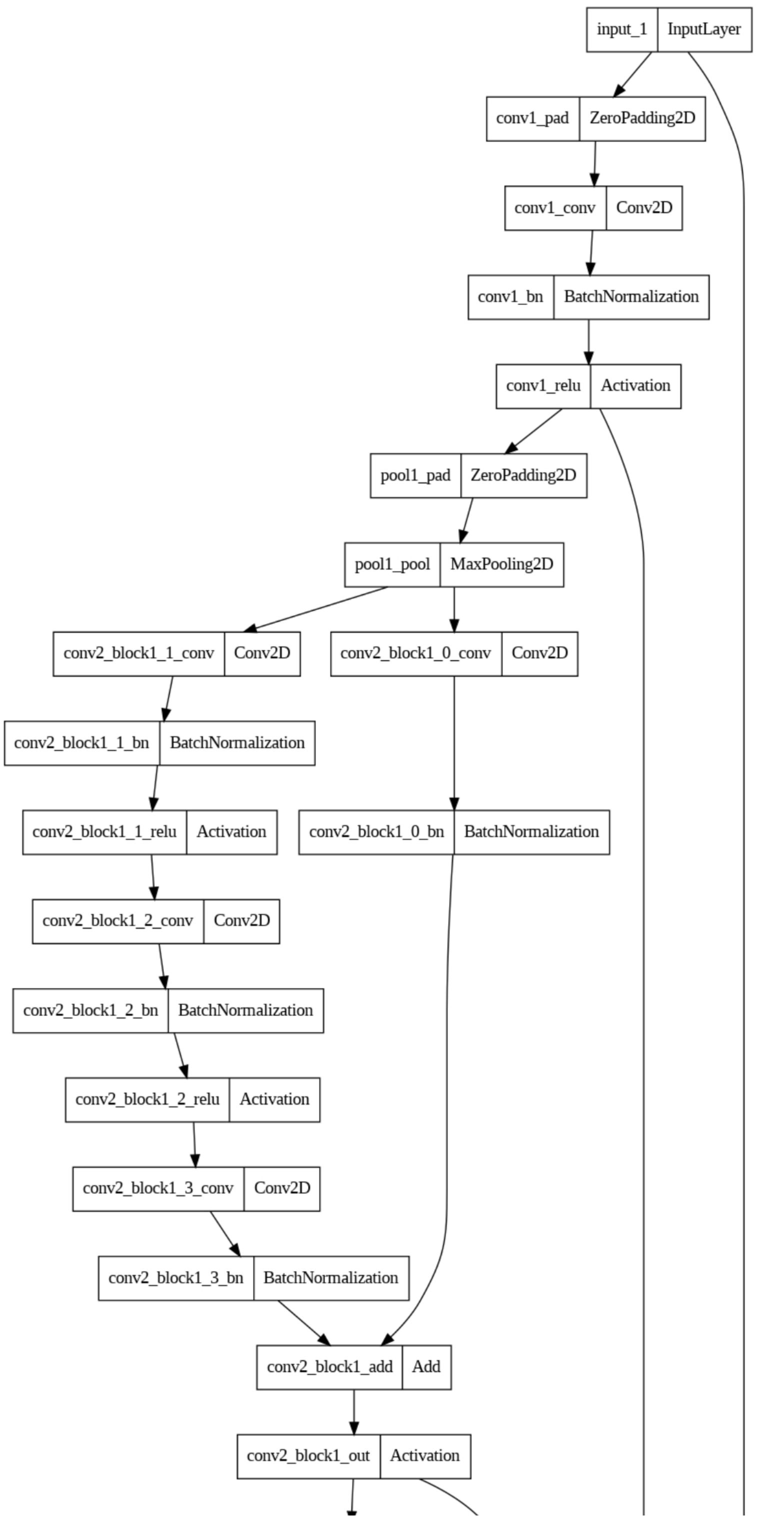
Column	Name	Type	Shape
0	input_1	InputLayer	[(None, 512, 512, 3)]
1	conv1_pad	ZeroPadding2D	(None, 518, 518, 3)
2	conv1_conv	Conv2D	(None, 256, 256, 64)
3	conv1_bn	BatchNormalization	(None, 256, 256, 64)
4	conv1_relu	Activation	(None, 256, 256, 64)
5	pool1_pad	ZeroPadding2D	(None, 258, 258, 64)
6	pool1_pool	MaxPooling2D	(None, 128, 128, 64)
7	conv2_block1_1_conv	Conv2D	(None, 128, 128, 64)
8	conv2_block1_1_bn	BatchNormalization	(None, 128, 128, 64)
9	conv2_block1_1_relu	Activation	(None, 128, 128, 64)
10	conv2_block1_2_conv	Conv2D	(None, 128, 128, 64)
11	conv2_block1_2_bn	BatchNormalization	(None, 128, 128, 64)
12	conv2_block1_2_relu	Activation	(None, 128, 128, 64)
13	conv2_block1_0_conv	Conv2D	(None, 128, 128, 256)
14	conv2_block1_3_conv	Conv2D	(None, 128, 128, 256)
15	conv2_block1_0_bn	BatchNormalization	(None, 128, 128, 256)
16	conv2_block1_3_bn	BatchNormalization	(None, 128, 128, 256)
17	conv2_block1_add	Add	(None, 128, 128, 256)
18	conv2_block1_out	Activation	(None, 128, 128, 256)
19	conv2_block2_1_conv	Conv2D	(None, 128, 128, 64)
20	conv2_block2_1_bn	BatchNormalization	(None, 128, 128, 64)
21	conv2_block2_1_relu	Activation	(None, 128, 128, 64)
22	conv2_block2_2_conv	Conv2D	(None, 128, 128, 64)
23	conv2_block2_2_bn	BatchNormalization	(None, 128, 128, 64)
24	conv2_block2_2_relu	Activation	(None, 128, 128, 64)
25	conv2_block2_3_conv	Conv2D	(None, 128, 128, 256)
26	conv2_block2_3_bn	BatchNormalization	(None, 128, 128, 256)
27	conv2_block2_add	Add	(None, 128, 128, 256)
28	conv2_block2_out	Activation	(None, 128, 128, 256)
29	conv2_block3_1_conv	Conv2D	(None, 128, 128, 64)
30	conv2_block3_1_bn	BatchNormalization	(None, 128, 128, 64)

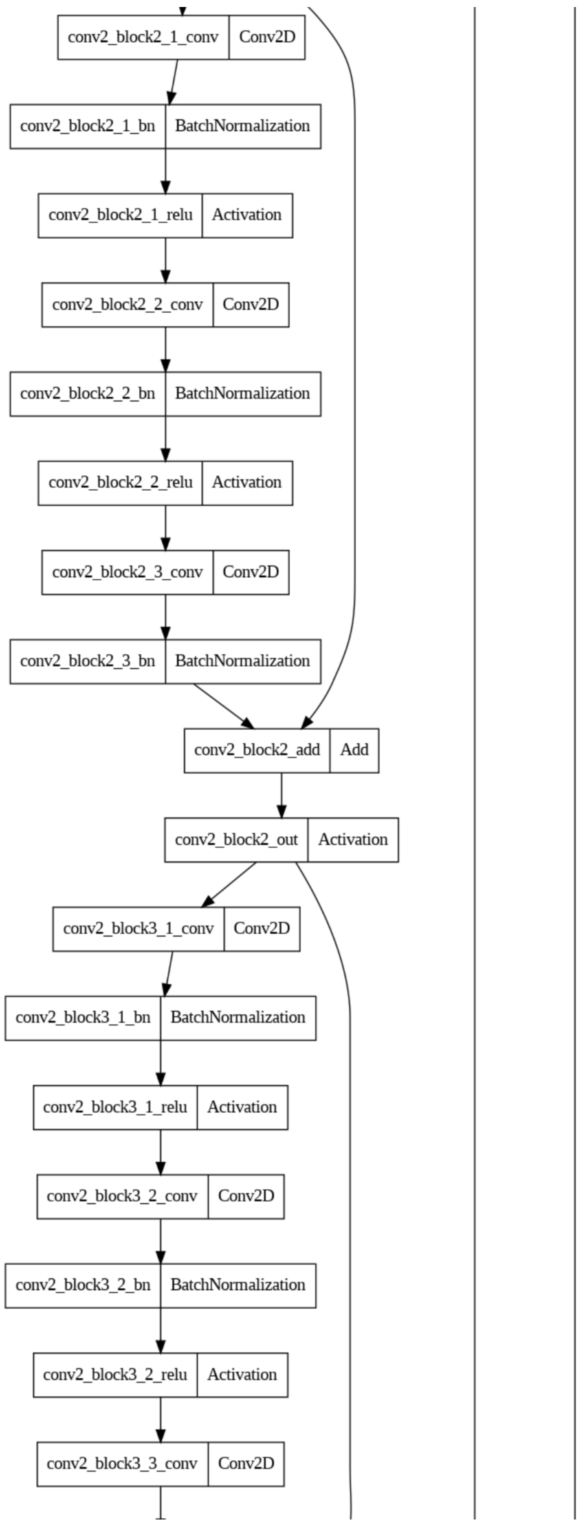
31	conv2_block3_1_relu	Activation	(None, 128, 128, 64)
32	conv2_block3_2_conv	Conv2D	(None, 128, 128, 64)
33	conv2_block3_2_bn	BatchNormalization	(None, 128, 128, 64)
34	conv2_block3_2_relu	Activation	(None, 128, 128, 64)
35	conv2_block3_3_conv	Conv2D	(None, 128, 128, 256)
36	conv2_block3_3_bn	BatchNormalization	(None, 128, 128, 256)
37	conv2_block3_add	Add	(None, 128, 128, 256)
38	conv2_block3_out	Activation	(None, 128, 128, 256)
39	conv3_block1_1_conv	Conv2D	(None, 64, 64, 128)
40	conv3_block1_1_bn	BatchNormalization	(None, 64, 64, 128)
41	conv3_block1_1_relu	Activation	(None, 64, 64, 128)
42	conv3_block1_2_conv	Conv2D	(None, 64, 64, 128)
43	conv3_block1_2_bn	BatchNormalization	(None, 64, 64, 128)
44	conv3_block1_2_relu	Activation	(None, 64, 64, 128)
45	conv3_block1_0_conv	Conv2D	(None, 64, 64, 512)
46	conv3_block1_3_conv	Conv2D	(None, 64, 64, 512)
47	conv3_block1_0_bn	BatchNormalization	(None, 64, 64, 512)
48	conv3_block1_3_bn	BatchNormalization	(None, 64, 64, 512)
49	conv3_block1_add	Add	(None, 64, 64, 512)
50	conv3_block1_out	Activation	(None, 64, 64, 512)
51	conv3_block2_1_conv	Conv2D	(None, 64, 64, 128)
52	conv3_block2_1_bn	BatchNormalization	(None, 64, 64, 128)
53	conv3_block2_1_relu	Activation	(None, 64, 64, 128)
54	conv3_block2_2_conv	Conv2D	(None, 64, 64, 128)
55	conv3_block2_2_bn	BatchNormalization	(None, 64, 64, 128)
56	conv3_block2_2_relu	Activation	(None, 64, 64, 128)
57	conv3_block2_3_conv	Conv2D	(None, 64, 64, 512)
58	conv3_block2_3_bn	BatchNormalization	(None, 64, 64, 512)
59	conv3_block2_add	Add	(None, 64, 64, 512)
60	conv3_block2_out	Activation	(None, 64, 64, 512)
61	conv3_block3_1_conv	Conv2D	(None, 64, 64, 128)
62	conv3_block3_1_bn	BatchNormalization	(None, 64, 64, 128)
63	conv3_block3_1_relu	Activation	(None, 64, 64, 128)
64	conv3_block3_2_conv	Conv2D	(None, 64, 64, 128)
65	conv3_block3_2_bn	BatchNormalization	(None, 64, 64, 128)
66	conv3_block3_2_relu	Activation	(None, 64, 64, 128)
67	conv3_block3_3_conv	Conv2D	(None, 64, 64, 512)
68	conv3_block3_3_bn	BatchNormalization	(None, 64, 64, 512)

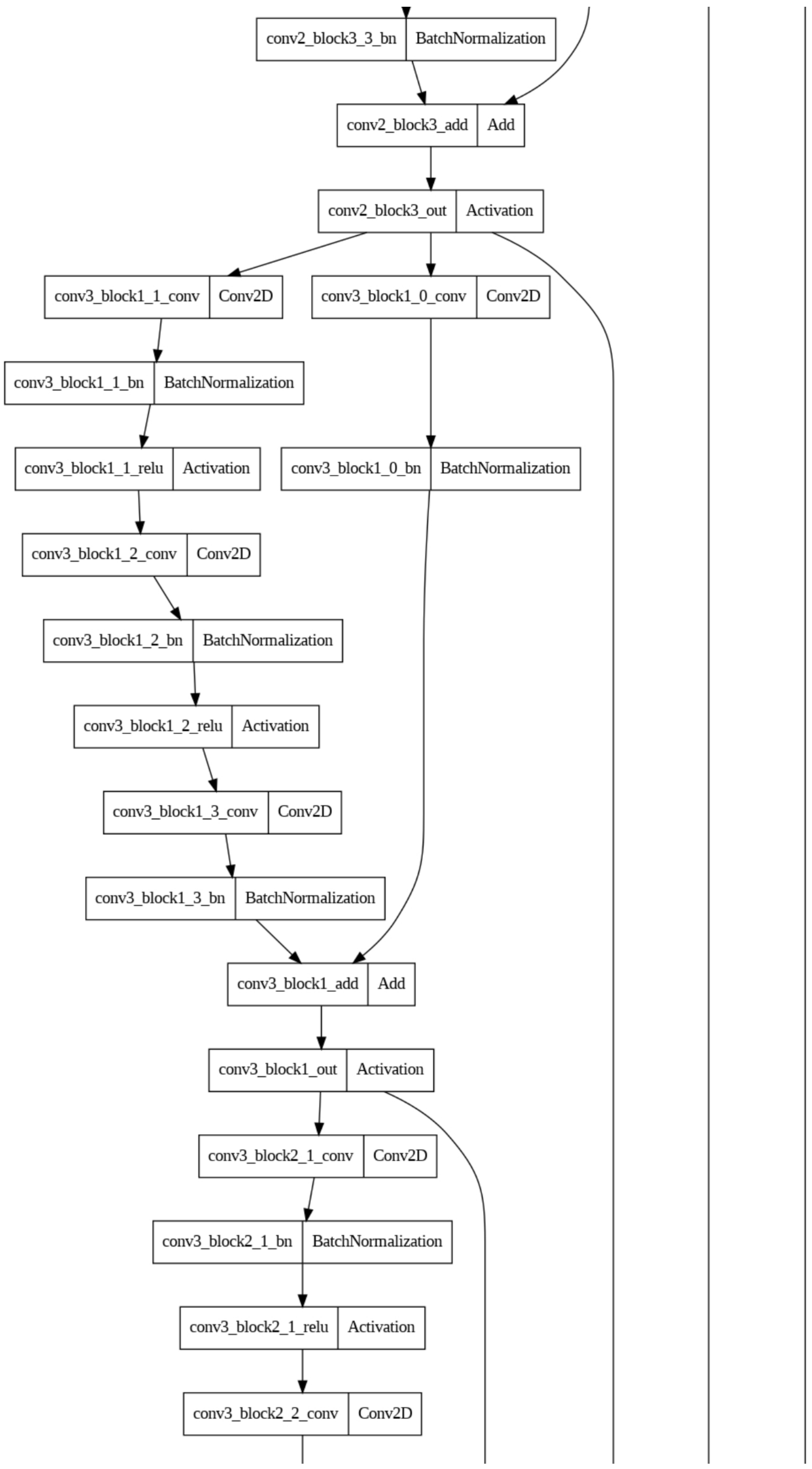
69	conv3_block3_add	Add	(None, 64, 64, 512)
70	conv3_block3_out	Activation	(None, 64, 64, 512)
71	conv3_block4_1_conv	Conv2D	(None, 64, 64, 128)
72	conv3_block4_1_bn	BatchNormalization	(None, 64, 64, 128)
73	conv3_block4_1_relu	Activation	(None, 64, 64, 128)
74	conv3_block4_2_conv	Conv2D	(None, 64, 64, 128)
75	conv3_block4_2_bn	BatchNormalization	(None, 64, 64, 128)
76	conv3_block4_2_relu	Activation	(None, 64, 64, 128)
77	conv3_block4_3_conv	Conv2D	(None, 64, 64, 512)
78	conv3_block4_3_bn	BatchNormalization	(None, 64, 64, 512)
79	conv3_block4_add	Add	(None, 64, 64, 512)
80	conv3_block4_out	Activation	(None, 64, 64, 512)
81	conv4_block1_1_conv	Conv2D	(None, 32, 32, 256)
82	conv4_block1_1_bn	BatchNormalization	(None, 32, 32, 256)
83	conv4_block1_1_relu	Activation	(None, 32, 32, 256)
84	conv4_block1_2_conv	Conv2D	(None, 32, 32, 256)
85	conv4_block1_2_bn	BatchNormalization	(None, 32, 32, 256)
86	conv4_block1_2_relu	Activation	(None, 32, 32, 256)
87	conv4_block1_0_conv	Conv2D	(None, 32, 32, 1024)
88	conv4_block1_3_conv	Conv2D	(None, 32, 32, 1024)
89	conv4_block1_0_bn	BatchNormalization	(None, 32, 32, 1024)
90	conv4_block1_3_bn	BatchNormalization	(None, 32, 32, 1024)
91	conv4_block1_add	Add	(None, 32, 32, 1024)
92	conv4_block1_out	Activation	(None, 32, 32, 1024)
93	conv4_block2_1_conv	Conv2D	(None, 32, 32, 256)
94	conv4_block2_1_bn	BatchNormalization	(None, 32, 32, 256)
95	conv4_block2_1_relu	Activation	(None, 32, 32, 256)
96	conv4_block2_2_conv	Conv2D	(None, 32, 32, 256)
97	conv4_block2_2_bn	BatchNormalization	(None, 32, 32, 256)
98	conv4_block2_2_relu	Activation	(None, 32, 32, 256)
99	conv4_block2_3_conv	Conv2D	(None, 32, 32, 1024)
100	conv4_block2_3_bn	BatchNormalization	(None, 32, 32, 1024)
101	conv4_block2_add	Add	(None, 32, 32, 1024)
102	conv4_block2_out	Activation	(None, 32, 32, 1024)
103	conv4_block3_1_conv	Conv2D	(None, 32, 32, 256)
104	conv4_block3_1_bn	BatchNormalization	(None, 32, 32, 256)
105	conv4_block3_1_relu	Activation	(None, 32, 32, 256)
106	conv4_block3_2_conv	Conv2D	(None, 32, 32, 256)

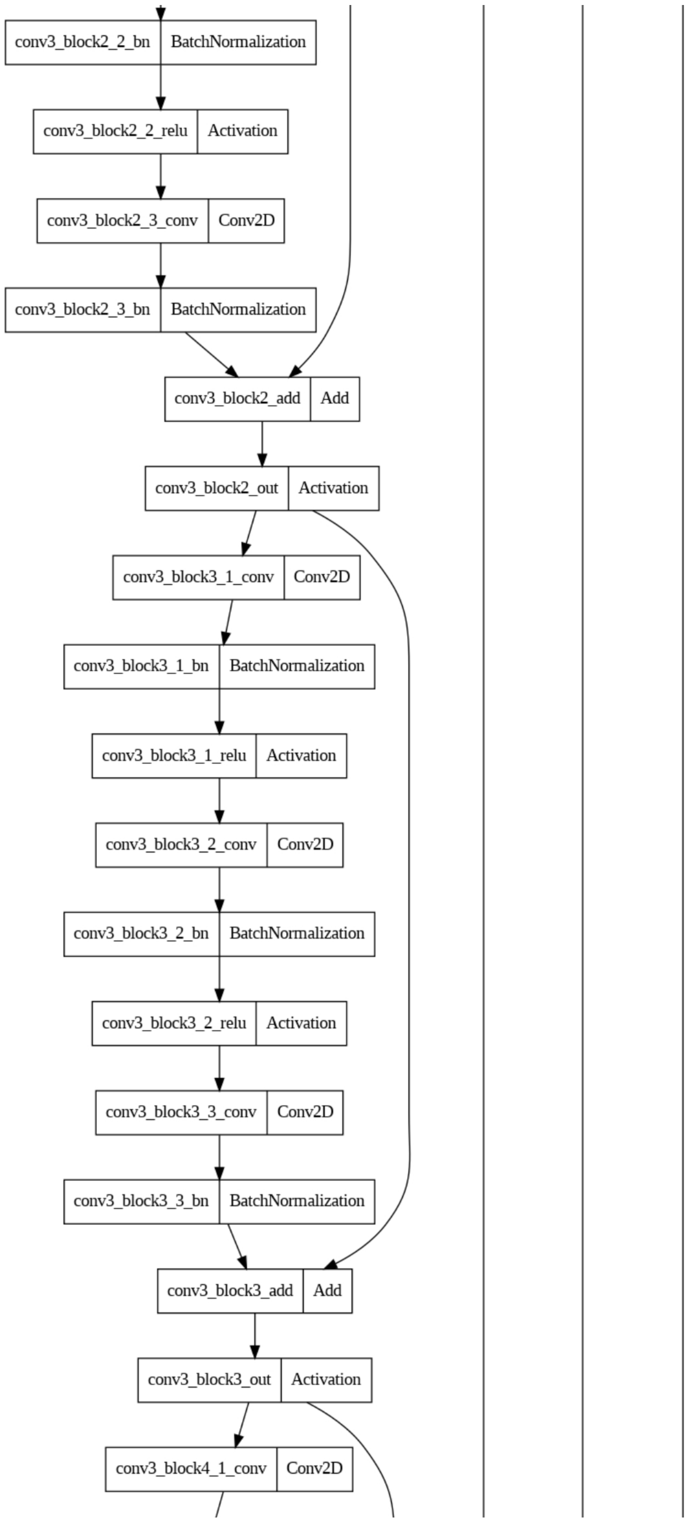
107	conv4_block3_2_bn	BatchNormalization	(None, 32, 32, 256)
108	conv4_block3_2_relu	Activation	(None, 32, 32, 256)
109	conv4_block3_3_conv	Conv2D	(None, 32, 32, 1024)
110	conv4_block3_3_bn	BatchNormalization	(None, 32, 32, 1024)
111	conv4_block3_add	Add	(None, 32, 32, 1024)
112	conv4_block3_out	Activation	(None, 32, 32, 1024)
113	conv4_block4_1_conv	Conv2D	(None, 32, 32, 256)
114	conv4_block4_1_bn	BatchNormalization	(None, 32, 32, 256)
115	conv4_block4_1_relu	Activation	(None, 32, 32, 256)
116	conv4_block4_2_conv	Conv2D	(None, 32, 32, 256)
117	conv4_block4_2_bn	BatchNormalization	(None, 32, 32, 256)
118	conv4_block4_2_relu	Activation	(None, 32, 32, 256)
119	conv4_block4_3_conv	Conv2D	(None, 32, 32, 1024)
120	conv4_block4_3_bn	BatchNormalization	(None, 32, 32, 1024)
121	conv4_block4_add	Add	(None, 32, 32, 1024)
122	conv4_block4_out	Activation	(None, 32, 32, 1024)
123	conv4_block5_1_conv	Conv2D	(None, 32, 32, 256)
124	conv4_block5_1_bn	BatchNormalization	(None, 32, 32, 256)
125	conv4_block5_1_relu	Activation	(None, 32, 32, 256)
126	conv4_block5_2_conv	Conv2D	(None, 32, 32, 256)
127	conv4_block5_2_bn	BatchNormalization	(None, 32, 32, 256)
128	conv4_block5_2_relu	Activation	(None, 32, 32, 256)
129	conv4_block5_3_conv	Conv2D	(None, 32, 32, 1024)
130	conv4_block5_3_bn	BatchNormalization	(None, 32, 32, 1024)
131	conv4_block5_add	Add	(None, 32, 32, 1024)
132	conv4_block5_out	Activation	(None, 32, 32, 1024)
133	conv4_block6_1_conv	Conv2D	(None, 32, 32, 256)
134	conv4_block6_1_bn	BatchNormalization	(None, 32, 32, 256)
135	conv4_block6_1_relu	Activation	(None, 32, 32, 256)
136	conv4_block6_2_conv	Conv2D	(None, 32, 32, 256)
137	conv4_block6_2_bn	BatchNormalization	(None, 32, 32, 256)
138	conv4_block6_2_relu	Activation	(None, 32, 32, 256)
139	conv4_block6_3_conv	Conv2D	(None, 32, 32, 1024)
140	conv4_block6_3_bn	BatchNormalization	(None, 32, 32, 1024)
141	conv4_block6_add	Add	(None, 32, 32, 1024)
142	conv4_block6_out	Activation	(None, 32, 32, 1024)
143	conv2d_transpose	Conv2DTranspose	(None, 64, 64, 512)
144	concatenate	Concatenate	(None, 64, 64, 1024)

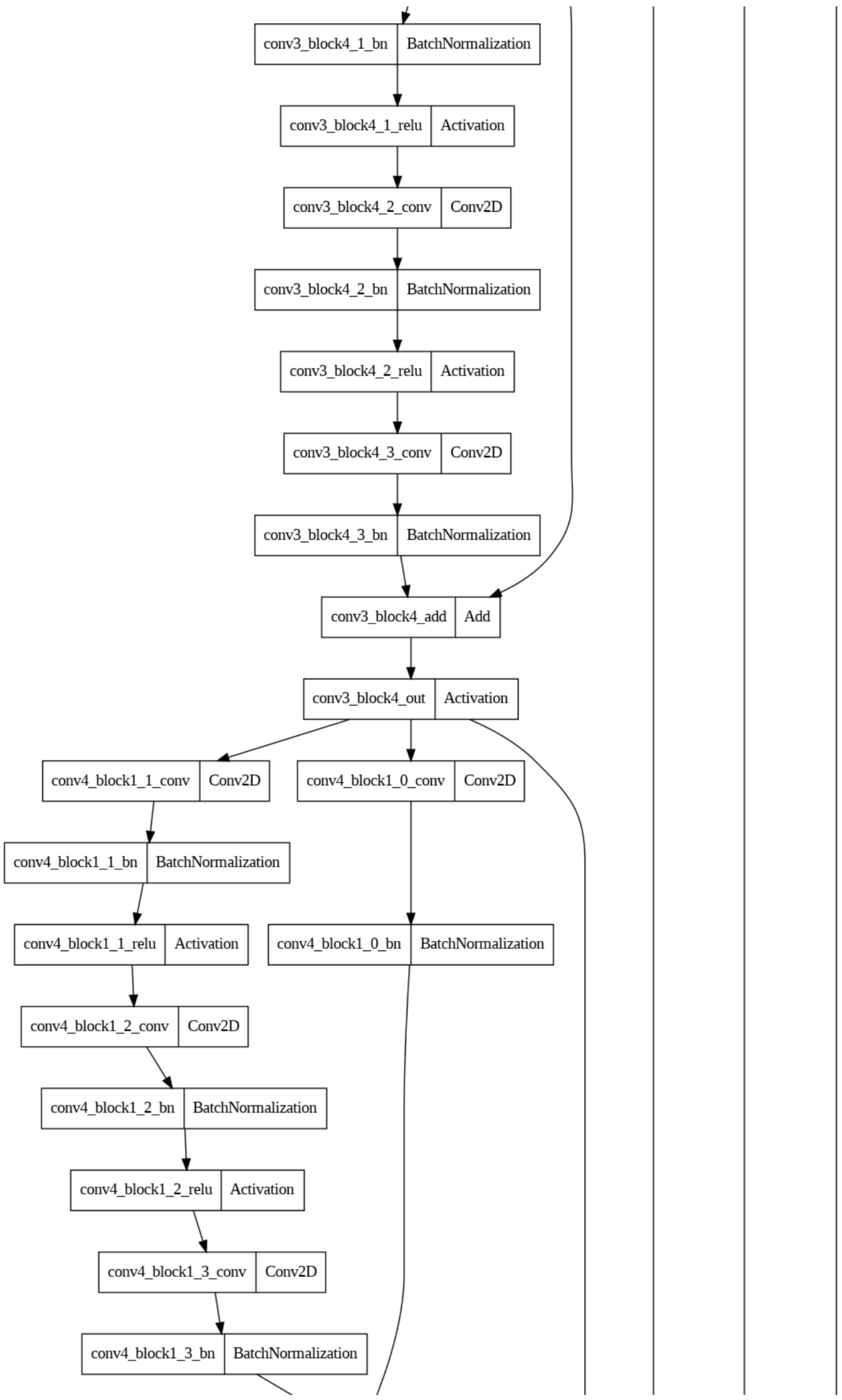
145	conv2d	Conv2D	(None, 64, 64, 512)
146	batch_normalization	BatchNormalization	(None, 64, 64, 512)
147	activation	Activation	(None, 64, 64, 512)
148	conv2d_1	Conv2D	(None, 64, 64, 512)
149	batch_normalization_1	BatchNormalization	(None, 64, 64, 512)
150	activation_1	Activation	(None, 64, 64, 512)
151	conv2d_transpose_1	Conv2DTranspose	(None, 128, 128, 256)
152	concatenate_1	Concatenate	(None, 128, 128, 512)
153	conv2d_2	Conv2D	(None, 128, 128, 256)
154	batch_normalization_2	BatchNormalization	(None, 128, 128, 256)
155	activation_2	Activation	(None, 128, 128, 256)
156	conv2d_3	Conv2D	(None, 128, 128, 256)
157	batch_normalization_3	BatchNormalization	(None, 128, 128, 256)
158	activation_3	Activation	(None, 128, 128, 256)
159	conv2d_transpose_2	Conv2DTranspose	(None, 256, 256, 128)
160	concatenate_2	Concatenate	(None, 256, 256, 192)
161	conv2d_4	Conv2D	(None, 256, 256, 128)
162	batch_normalization_4	BatchNormalization	(None, 256, 256, 128)
163	activation_4	Activation	(None, 256, 256, 128)
164	conv2d_5	Conv2D	(None, 256, 256, 128)
165	batch_normalization_5	BatchNormalization	(None, 256, 256, 128)
166	activation_5	Activation	(None, 256, 256, 128)
167	conv2d_transpose_3	Conv2DTranspose	(None, 512, 512, 64)
168	concatenate_3	Concatenate	(None, 512, 512, 67)
169	conv2d_6	Conv2D	(None, 512, 512, 64)
170	batch_normalization_6	BatchNormalization	(None, 512, 512, 64)
171	activation_6	Activation	(None, 512, 512, 64)
172	conv2d_7	Conv2D	(None, 512, 512, 64)
173	batch_normalization_7	BatchNormalization	(None, 512, 512, 64)
174	activation_7	Activation	(None, 512, 512, 64)
175	conv2d_8	Conv2D	(None, 512, 512, 1)

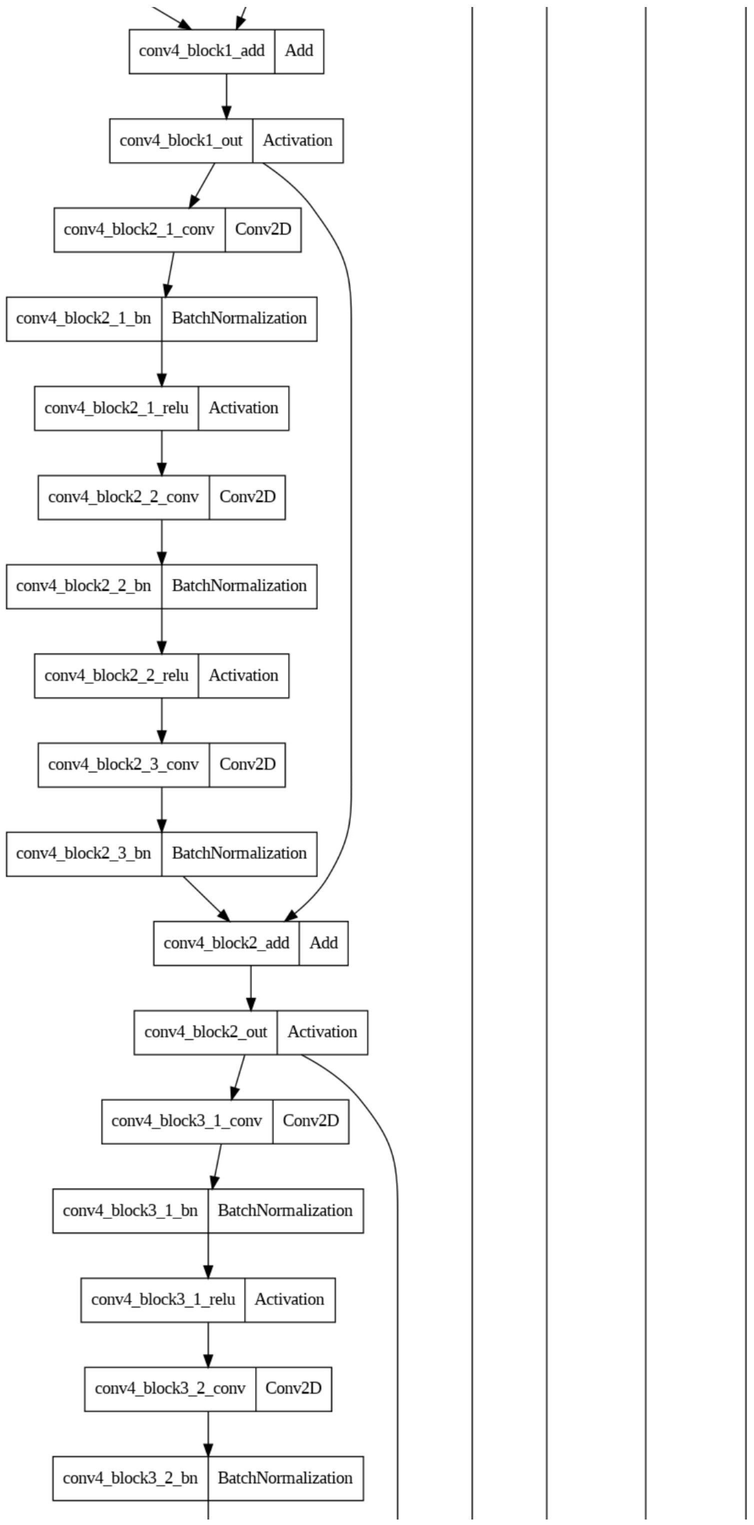

input_1 InputLayer
conv1_pad ZeroPadding2D
conv1_conv Conv2D
conv1_bn BatchNormalization
conv1_relu Activation
pool1_pad ZeroPadding2D
pool1_pool MaxPooling2D
conv2_block1_1_conv Conv2D
conv2_block1_0_conv Conv2D
conv2_block1_1_bn BatchNormalization
conv2_block1_1_relu Activation
conv2_block1_0_bn BatchNormalization
conv2_block1_2_conv Conv2D
conv2_block1_2_bn BatchNormalization
conv2_block1_2_relu Activation
conv2_block1_3_conv Conv2D
conv2_block1_3_bn BatchNormalization
conv2_block1_add Add
conv2_block1_out Activation

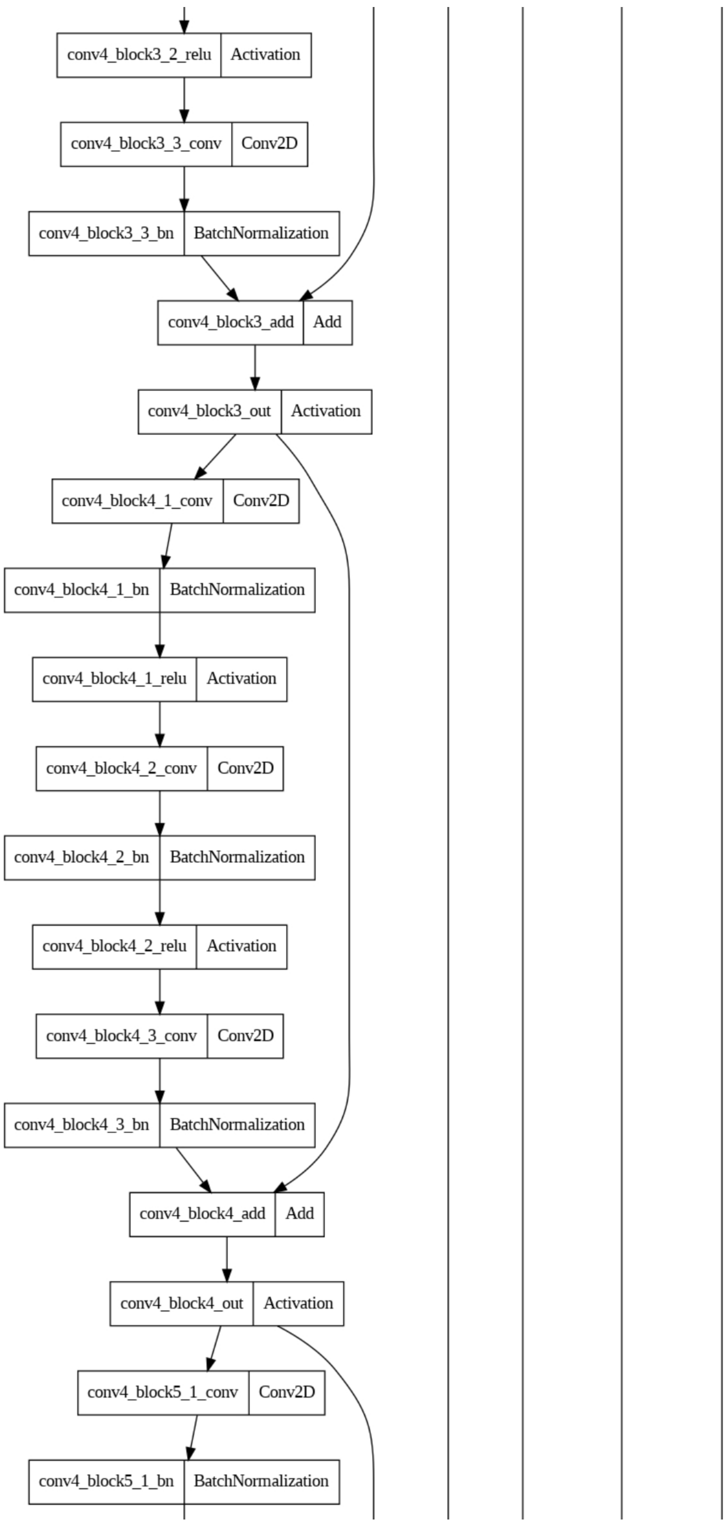

conv2_block2_1_conv | Conv2D
conv2_block2_1_bn | BatchNormalization
conv2_block2_1_relu | Activation
conv2_block2_2_conv | Conv2D
conv2_block2_2_bn | BatchNormalization
conv2_block2_2_relu | Activation
conv2_block2_3_conv | Conv2D
conv2_block2_3_bn | BatchNormalization
conv2_block2_add | Add
conv2_block2_out | Activation
conv2_block3_1_conv | Conv2D
conv2_block3_1_bn | BatchNormalization
conv2_block3_1_relu | Activation
conv2_block3_2_conv | Conv2D
conv2_block3_2_bn | BatchNormalization
conv2_block3_2_relu | Activation
conv2_block3_3_conv | Conv2D

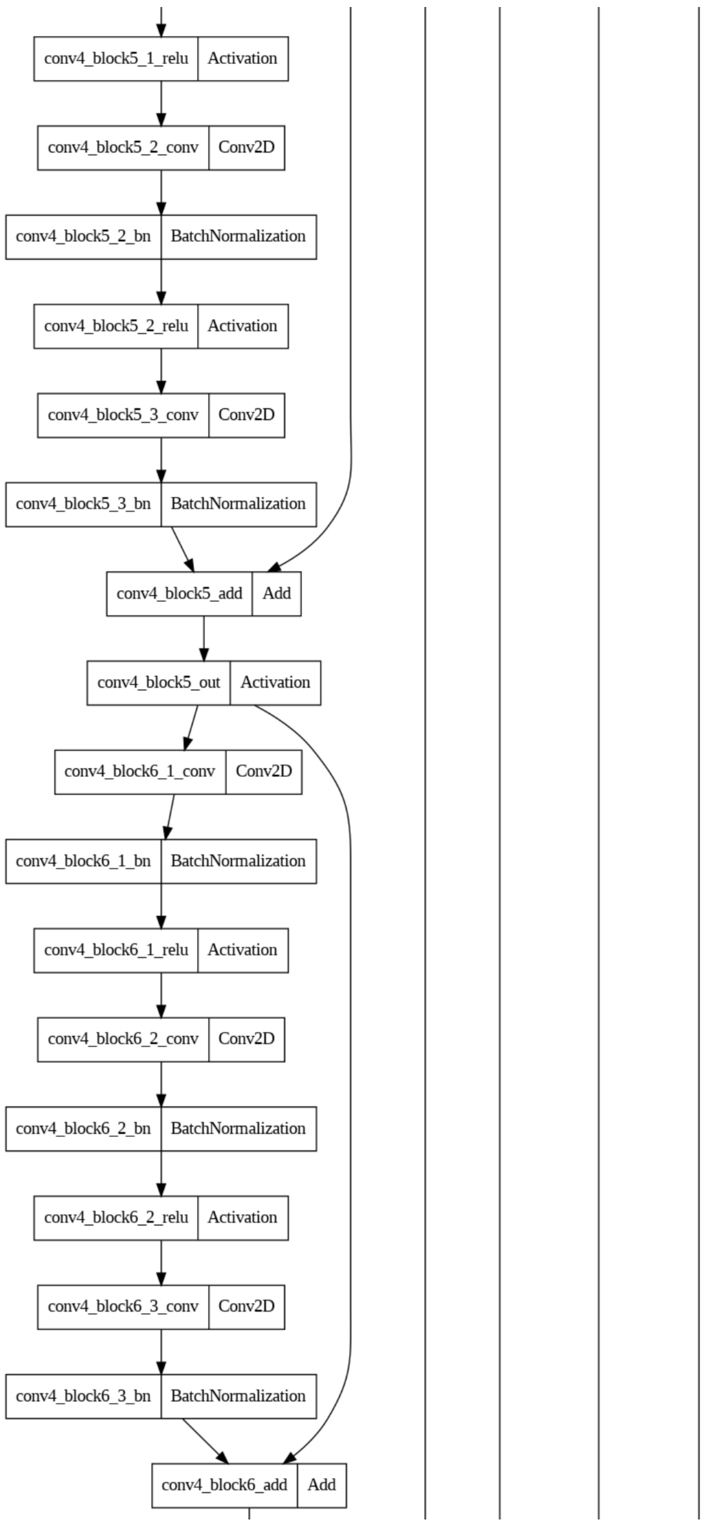

conv2_block3_3_bn | BatchNormalization
conv2_block3_add | Add
conv2_block3_out | Activation
conv3_block1_1_conv | Conv2D
conv3_block1_0_conv | Conv2D
conv3_block1_1_bn | BatchNormalization
conv3_block1_1_relu | Activation
conv3_block1_0_bn | BatchNormalization
conv3_block1_2_conv | Conv2D
conv3_block1_2_bn | BatchNormalization
conv3_block1_2_relu | Activation
conv3_block1_3_conv | Conv2D
conv3_block1_3_bn | BatchNormalization
conv3_block1_add | Add
conv3_block1_out | Activation
conv3_block2_1_conv | Conv2D
conv3_block2_1_bn | BatchNormalization
conv3_block2_1_relu | Activation
conv3_block2_2_conv | Conv2D

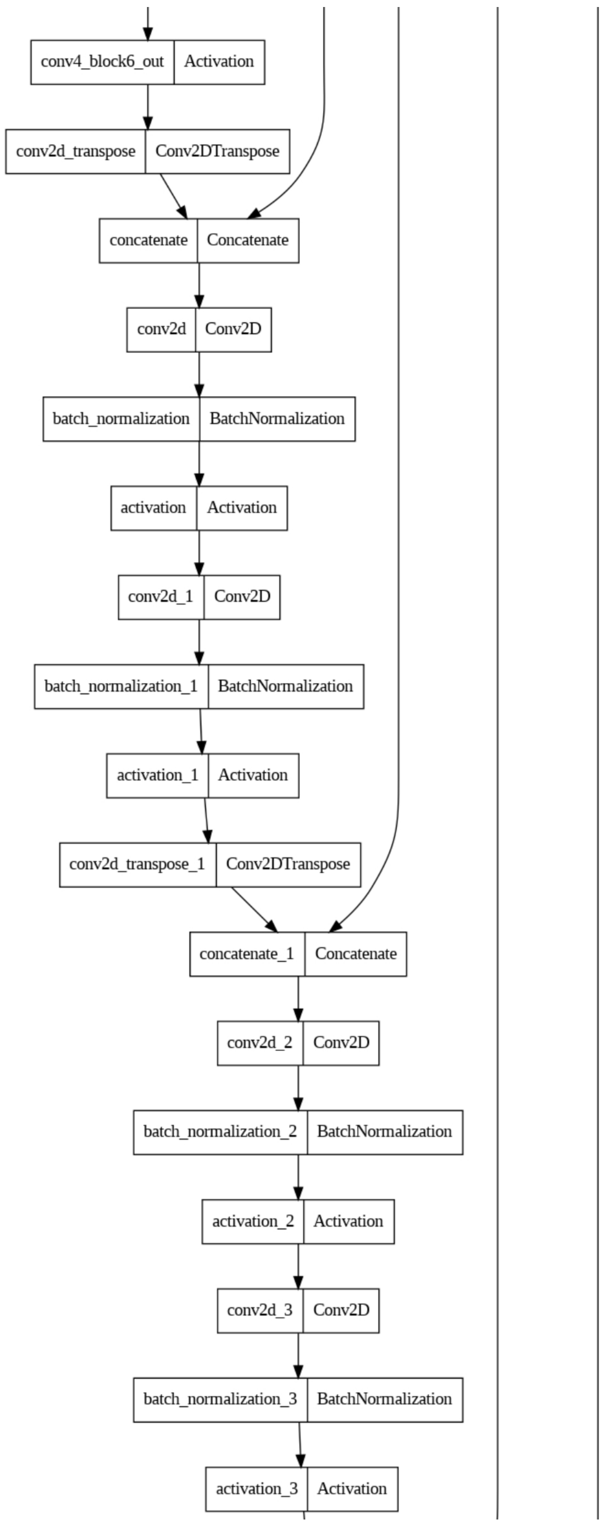

conv3_block2_2_bn | BatchNormalization
conv3_block2_2_relu | Activation
conv3_block2_3_conv | Conv2D
conv3_block2_3_bn | BatchNormalization
conv3_block2_add | Add
conv3_block2_out | Activation
conv3_block3_1_conv | Conv2D
conv3_block3_1_bn | BatchNormalization
conv3_block3_1_relu | Activation
conv3_block3_2_conv | Conv2D
conv3_block3_2_bn | BatchNormalization
conv3_block3_2_relu | Activation
conv3_block3_3_conv | Conv2D
conv3_block3_3_bn | BatchNormalization
conv3_block3_add | Add
conv3_block3_out | Activation
conv3_block4_1_conv | Conv2D


conv3_block4_1_bn | BatchNormalization
conv3_block4_1_relu | Activation
conv3_block4_2_conv | Conv2D
conv3_block4_2_bn | BatchNormalization
conv3_block4_2_relu | Activation
conv3_block4_3_conv | Conv2D
conv3_block4_3_bn | BatchNormalization
conv3_block4_add | Add
conv3_block4_out | Activation
conv4_block1_1_conv | Conv2D
conv4_block1_0_conv | Conv2D
conv4_block1_1_bn | BatchNormalization
conv4_block1_1_relu | Activation
conv4_block1_0_bn | BatchNormalization
conv4_block1_2_conv | Conv2D
conv4_block1_2_bn | BatchNormalization
conv4_block1_2_relu | Activation
conv4_block1_3_conv | Conv2D
conv4_block1_3_bn | BatchNormalization


conv4_block1_add | Add
conv4_block1_out | Activation
conv4_block2_1_conv | Conv2D
conv4_block2_1_bn | BatchNormalization
conv4_block2_1_relu | Activation
conv4_block2_2_conv | Conv2D
conv4_block2_2_bn | BatchNormalization
conv4_block2_2_relu | Activation
conv4_block2_3_conv | Conv2D
conv4_block2_3_bn | BatchNormalization
conv4_block2_add | Add
conv4_block2_out | Activation
conv4_block3_1_conv | Conv2D
conv4_block3_1_bn | BatchNormalization
conv4_block3_1_relu | Activation
conv4_block3_2_conv | Conv2D
conv4_block3_2_bn | BatchNormalization


conv4_block3_2_relu | Activation
conv4_block3_3_conv | Conv2D
conv4_block3_3_bn | BatchNormalization
conv4_block3_add | Add
conv4_block3_out | Activation
conv4_block4_1_conv | Conv2D
conv4_block4_1_bn | BatchNormalization
conv4_block4_1_relu | Activation
conv4_block4_2_conv | Conv2D
conv4_block4_2_bn | BatchNormalization
conv4_block4_2_relu | Activation
conv4_block4_3_conv | Conv2D
conv4_block4_3_bn | BatchNormalization
conv4_block4_add | Add
conv4_block4_out | Activation
conv4_block5_1_conv | Conv2D
conv4_block5_1_bn | BatchNormalization


conv4_block5_1_relu | Activation
conv4_block5_2_conv | Conv2D
conv4_block5_2_bn | BatchNormalization
conv4_block5_2_relu | Activation
conv4_block5_3_conv | Conv2D
conv4_block5_3_bn | BatchNormalization
conv4_block5_add | Add
conv4_block5_out | Activation
conv4_block6_1_conv | Conv2D
conv4_block6_1_bn | BatchNormalization
conv4_block6_1_relu | Activation
conv4_block6_2_conv | Conv2D
conv4_block6_2_bn | BatchNormalization
conv4_block6_2_relu | Activation
conv4_block6_3_conv | Conv2D
conv4_block6_3_bn | BatchNormalization
conv4_block6_add | Add


conv4_block6_out | Activation
conv2d_transpose | Conv2DTranspose
concatenate | Concatenate
conv2d | Conv2D
batch_normalization | BatchNormalization
activation | Activation
conv2d_1 | Conv2D
batch_normalization_1 | BatchNormalization
activation_1 | Activation
conv2d_transpose_1 | Conv2DTranspose
concatenate_1 | Concatenate
conv2d_2 | Conv2D
batch_normalization_2 | BatchNormalization
activation_2 | Activation
conv2d_3 | Conv2D
batch_normalization_3 | BatchNormalization
activation_3 | Activation

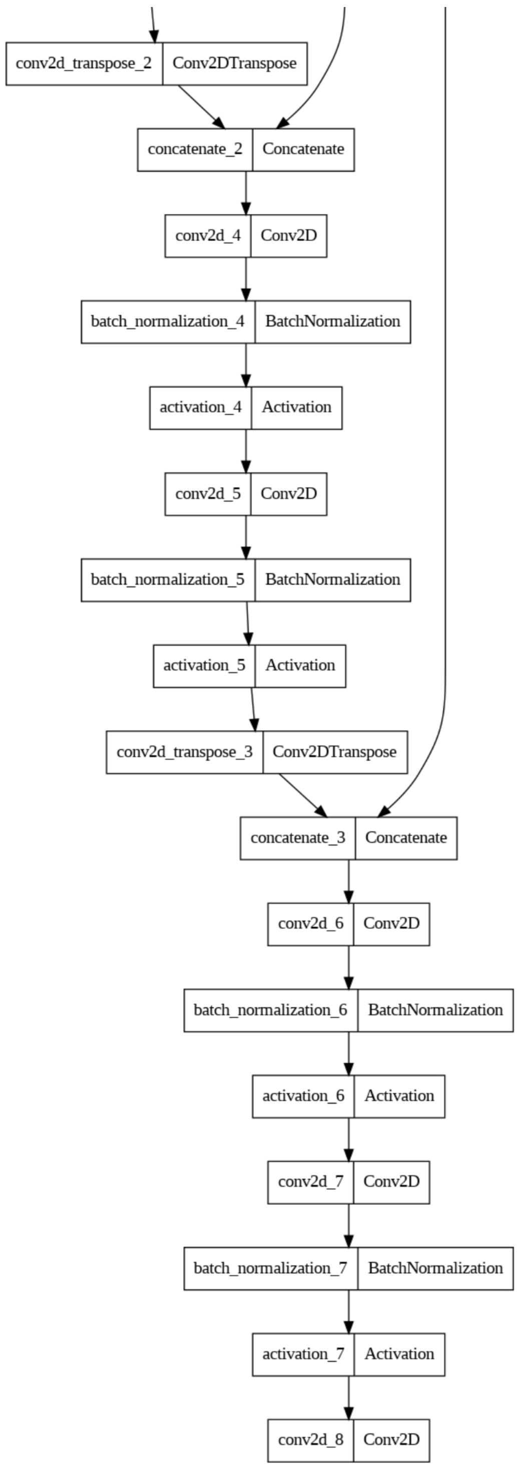

conv2d_transpose_2 | Conv2DTranspose
concatenate_2 | Concatenate
conv2d_4 | Conv2D
batch_normalization_4 | BatchNormalization
activation_4 | Activation
conv2d_5 | Conv2D
batch_normalization_5 | BatchNormalization
activation_5 | Activation
conv2d_transpose_3 | Conv2DTranspose
concatenate_3 | Concatenate
conv2d_6 | Conv2D
batch_normalization_6 | BatchNormalization
activation_6 | Activation
conv2d_7 | Conv2D
batch_normalization_7 | BatchNormalization
activation_7 | Activation
conv2d_8 | Conv2D

Figure 5.22 Graph representation of the model

References

1. Ragnoli, A.; De Blasiis, M.; Di Benedetto, A.; Blasiis, M.D.; Benedetto, A.D. Pavement Distress Detection Methods: A Review. Infrastructures, 3 (4), 58. **2018**.
2. Barbarella, M.; Di Benedetto, A.; Fiani, M. A Method for Obtaining a DEM with Curved Abscissa from MLS Data for Linear Infrastructure Survey Design. *Remote Sensing* **2022**, *14*, 889.
3. Zhang, L.; Yang, F.; Zhang, Y.D.; Zhu, Y.J. Road crack detection using deep convolutional neural network. In Proceedings of the 2016 IEEE international conference on image processing (ICIP), 2016; pp. 3708-3712.
4. Wang, W.; Wang, M.; Li, H.; Zhao, H.; Wang, K.; He, C.; Wang, J.; Zheng, S.; Chen, J. Pavement crack image acquisition methods and crack extraction algorithms: A review. *Journal of Traffic and Transportation Engineering (English Edition)* **2019**, *6*, 535-556.
5. Falkingham, P.L. Acquisition of high resolution three-dimensional models using free, open-source, photogrammetric software. *Palaeontologia Electronica* **2012**, *15*.
6. Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; Fei-Fei, L. Large-scale video classification with convolutional neural networks. In Proceedings of the Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2014; pp. 1725-1732.
7. Yu, S.; Jia, S.; Xu, C. Convolutional neural networks for hyperspectral image classification. *Neurocomputing* **2017**, *219*, 88-98.
8. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *nature* **2015**, *521*, 436-444.
9. Russell, S.J. *Artificial intelligence a modern approach*; Pearson Education, Inc.: 2010.
10. Mitchell, T.M. *Machine learning*; McGraw-hill New York: 2007; Volume 1.
11. Bishop, C.M.; Nasrabadi, N.M. *Pattern recognition and machine learning*; Springer: 2006; Volume 4.
12. Cortes, C.; Vapnik, V. Support-vector networks. *Machine learning* **1995**, *20*, 273-297.
13. Quinlan, J.R. Induction of decision trees. *Machine learning* **1986**, *1*, 81-106.
14. Breiman, L. Random forests. *Machine learning* **2001**, *45*, 5-32.
15. Friedman, J.; Hastie, T.; Tibshirani, R. *The elements of statistical learning*; Springer series in statistics New York: 2001; Volume 1.
16. MacQueen, J. Classification and analysis of multivariate observations. In Proceedings of the 5th Berkeley Symp. Math. Statist. Probability, 1967; pp. 281-297.
17. Lloyd, S. Least squares quantization in PCM. *IEEE transactions on information theory* **1982**, *28*, 129-137.
18. Johnson, S.C. Hierarchical clustering schemes. *Psychometrika* **1967**, *32*, 241-254.

19. Abdi, H.; Williams, L.J. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics* **2010**, *2*, 433-459.
20. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *Journal of machine learning research* **2008**, *9*.
21. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. *Journal of artificial intelligence research* **1996**, *4*, 237-285.
22. Watkins, C.J.; Dayan, P. Q-learning. *Machine learning* **1992**, *8*, 279-292.
23. Sutton, R.S.; McAllester, D.; Singh, S.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* **1999**, *12*.
24. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *nature* **1986**, *323*, 533-536.
25. Mikolov, T.; Karafiát, M.; Burget, L.; Cernocký, J.; Khudanpur, S. Recurrent neural network based language model. In Proceedings of the Interspeech, 2010; pp. 1045-1048.
26. Graves, A.; Mohamed, A.-r.; Hinton, G. Speech recognition with deep recurrent neural networks. In Proceedings of the 2013 IEEE international conference on acoustics, speech and signal processing, 2013; pp. 6645-6649.
27. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural computation* **1997**, *9*, 1735-1780.
28. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* **1994**, *5*, 157-166.
29. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation applied to handwritten zip code recognition. *Neural computation* **1989**, *1*, 541-551.
30. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* **2012**, *25*.
31. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* **2014**.
32. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2016; pp. 770-778.
33. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **1998**, *86*, 2278-2324.
34. Dumoulin, V.; Visin, F. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285* **2016**.

35. Boureau, Y.-L.; Ponce, J.; LeCun, Y. A theoretical analysis of feature pooling in visual recognition. In Proceedings of the Proceedings of the 27th international conference on machine learning (ICML-10), 2010; pp. 111-118.

36. Scherer, D.; Müller, A.; Behnke, S. Evaluation of pooling operations in convolutional architectures for object recognition. In Proceedings of the Artificial Neural Networks–ICANN 2010: 20th International Conference, Thessaloniki, Greece, September 15-18, 2010, Proceedings, Part III 20, 2010; pp. 92-101.

37. Hahnloser, R.H.; Sarpeshkar, R.; Mahowald, M.A.; Douglas, R.J.; Seung, H.S. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *nature* **2000**, *405*, 947-951.

38. Glorot, X.; Bordes, A.; Bengio, Y. Deep sparse rectifier neural networks. In Proceedings of the Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011; pp. 315-323.

39. Han, J.; Moraga, C. The influence of the sigmoid function parameters on the speed of backpropagation learning. In Proceedings of the From Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks Malaga-Torremolinos, Spain, June 7–9, 1995 Proceedings 3, 1995; pp. 195-201.

40. Namin, A.H.; Leboeuf, K.; Muscedere, R.; Wu, H.; Ahmadi, M. Efficient hardware implementation of the hyperbolic tangent sigmoid function. In Proceedings of the 2009 IEEE International Symposium on Circuits and Systems, 2009; pp. 2117-2120.

41. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2015; pp. 1-9.

42. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical image computing and computer-assisted intervention, 2015; pp. 234-241.

43. Zou, Q.; Cao, Y.; Li, Q.; Mao, Q.; Wang, S. CrackTree: Automatic crack detection from pavement images. *Pattern Recognition Letters* **2012**, *33*, 227-238.

44. Dung, C.V. Autonomous concrete crack detection using deep fully convolutional neural network. *Automation in Construction* **2019**, *99*, 52-58.

45. Oliveira, H.; Correia, P.L. Automatic road crack detection and characterization. *IEEE Transactions on Intelligent Transportation Systems* **2012**, *14*, 155-168.

46. Koch, C.; Georgieva, K.; Kasireddy, V.; Akinci, B.; Fieguth, P. A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure. *Advanced Engineering Informatics* **2015**, *29*, 196-210.

47. Sansalone, M.; Sansalone, M.; Carino, N.J. *Impact-echo: A method for flaw detection in concrete using transient stress waves*; United States Department of Commerce, National Institute of Standards and …: 1986.
48. Wolf, J.; Pirskawetz, S.; Zang, A. Detection of crack propagation in concrete with embedded ultrasonic sensors. *Engineering Fracture Mechanics* **2015**, *146*, 161-171.
49. Shiotani, T.; Ohtsu, M.; Ikeda, K. Detection and evaluation of AE waves due to rock deformation. *Construction and Building Materials* **2001**, *15*, 235-246.
50. Yuyama, S.; Ohtsu, M. Acoustic emission evaluation in concrete. *Acoustic Emission-Beyond the Millennium; Kishi, T., Ohtsu, M., Yuyama, S., Eds* **2000**, 187-213.
51. Daniels, D.J. *Ground penetrating radar*; Iet: 2004; Volume 1.
52. Barnes, C.L.; Trottier, J.-F. Ground-penetrating radar for network-level concrete deck repair management. *Journal of Transportation Engineering* **2000**, *126*, 257-262.
53. Birtwisle, A.; Utsi, E. The use of ground penetrating radar to detect vertical subsurface cracking in airport runways. In Proceedings of the Proceedings of the 12th International Conference on Ground Penetrating Radar (GPR2008), Birmingham, UK, 2008.
54. Fröhlich, C.; Mettenleiter, M. Terrestrial laser scanning–new perspectives in 3D surveying. *International archives of photogrammetry, remote sensing and spatial information sciences* **2004**, *36*, W2.
55. Vosselman, G.; Maas, H.-G. *Airborne and terrestrial laser scanning*; CRC press: 2010.
56. Soudarissanane, S.; Lindenbergh, R.; Menenti, M.; Teunissen, P. Scanning geometry: Influencing factor on the quality of terrestrial laser scanning points. *ISPRS journal of photogrammetry and remote sensing* **2011**, *66*, 389-399.
57. Kashani, A.G.; Olsen, M.J.; Parrish, C.E.; Wilson, N. A review of LiDAR radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration. *Sensors* **2015**, *15*, 28099-28128.
58. Truong-Hong, L.; Laefer, D.F. Application of terrestrial laser scanner in bridge inspection: review and an opportunity. In Proceedings of the 37th IABSE Symposium: Engineering for Progress, Nature and People, Madrid, Spain, 3-5 September 2014, 2014.
59. Suchocki, C. Comparison of time-of-flight and phase-shift TLS intensity data for the diagnostics measurements of buildings. *Materials* **2020**, *13*, 353.
60. Laurent, J.; Lefebvre, D.; Samson, E. Development of a new 3D transverse laser profiling system for the automatic measurement of road cracks. In Proceedings of the Symposium on Pavement Surface Characteristics, 6th, 2008, Portoroz, Slovenia, 2008.

61. Hoensheid, R.C.; Dean, D.B.; Brooks, C.; Ahlborn, T.M.; Harris, D.K.; Dobson, R.J. *An Evaluation of Surface Defect Detection in Reinforced Concrete Bridge Decks Using Terrestrial LiDAR*; 2012.
62. Williams, K.; Olsen, M.J.; Roe, G.V.; Glennie, C. Synthesis of transportation applications of mobile LiDAR. *Remote Sensing* **2013**, *5*, 4652-4692.
63. Guan, H.; Li, J.; Yu, Y.; Chapman, M.; Wang, H.; Wang, C.; Zhai, R. Iterative tensor voting for pavement crack extraction using mobile laser scanning data. *IEEE Transactions on Geoscience and Remote Sensing* **2014**, *53*, 1527-1537.
64. Wang, P.; Huang, H. Comparison analysis on present image-based crack detection methods in concrete structures. In Proceedings of the 2010 3rd international congress on image and signal processing, 2010; pp. 2530-2533.
65. Ko, J.; Ni, Y.; Zhou, H.; Wang, J.; Zhou, X. Investigation concerning structural health monitoring of an instrumented cable-stayed bridge. *Structures & Infrastructure Engineering* **2009**, *5*, 497-513.
66. Jahanshahi, M.R.; Masri, S.F.; Sukhatme, G.S. Multi-image stitching and scene reconstruction for evaluating defect evolution in structures. *Structural Health Monitoring* **2011**, *10*, 643-657.
67. Ho, H.-N.; Kim, K.-D.; Park, Y.-S.; Lee, J.-J. An efficient image-based damage detection for cable surface in cable-stayed bridges. *Ndt & E International* **2013**, *58*, 18-23.
68. Jiang, S.; Zhang, J. Real-time crack assessment using deep neural networks with wall-climbing unmanned aerial system. *Computer-Aided Civil and Infrastructure Engineering* **2020**, *35*, 549-564.
69. Yeum, C.M.; Dyke, S.J. Vision-based automated crack detection for bridge inspection. *Computer-Aided Civil and Infrastructure Engineering* **2015**, *30*, 759-770.
70. Tanaka, N.; Uematsu, K. A Crack Detection Method in Road Surface Images Using Morphology. *MVA* **1998**, *98*, 17-19.
71. Iyer, S.; Sinha, S.K. A robust approach for automatic detection and segmentation of cracks in underground pipeline images. *Image and Vision Computing* **2005**, *23*, 921-933.
72. Yiyang, Z. The design of glass crack detection system based on image preprocessing technology. In Proceedings of the 2014 IEEE 7th joint international information technology and artificial intelligence conference, 2014; pp. 39-42.
73. Talab, A.M.A.; Huang, Z.; Xi, F.; HaiMing, L. Detection crack in image using Otsu method and multiple filtering in image processing techniques. *Optik* **2016**, *127*, 1030-1033.

74. Salman, M.; Mathavan, S.; Kamal, K.; Rahman, M. Pavement crack detection using the Gabor filter. In Proceedings of the 16th international IEEE conference on intelligent transportation systems (ITSC 2013), 2013; pp. 2039-2044.
75. Huang, Y.; Xu, B. Automatic inspection of pavement cracking distress. *Journal of Electronic Imaging* **2006**, *15*, 013017-013017-013016.
76. Zhao, H.; Qin, G.; Wang, X. Improvement of canny algorithm based on pavement edge detection. In Proceedings of the 2010 3rd international congress on image and signal processing, 2010; pp. 964-967.
77. Abdel-Qader, I.; Abudayyeh, O.; Kelly, M.E. Analysis of edge-detection techniques for crack identification in bridges. *Journal of Computing in Civil Engineering* **2003**, *17*, 255-263.
78. Xie, S.; Tu, Z. Holistically-nested edge detection. In Proceedings of the Proceedings of the IEEE international conference on computer vision, 2015; pp. 1395-1403.
79. Lattanzi, D.; Miller, G.R. Robust automated concrete damage detection algorithms for field applications. *Journal of Computing in Civil Engineering* **2014**, *28*, 253-262.
80. Zhang, W.; Zhang, Z.; Qi, D.; Liu, Y. Automatic crack detection and classification method for subway tunnel safety monitoring. *Sensors* **2014**, *14*, 19307-19328.
81. Moussa, G.; Hussain, K. A new technique for automatic detection and parameters estimation of pavement crack. In Proceedings of the 4th International Multi-Conference on Engineering Technology Innovation, IMETI, 2011.
82. Prasanna, P.; Dana, K.J.; Gucunski, N.; Basily, B.B.; La, H.M.; Lim, R.S.; Parvardeh, H. Automated crack detection on concrete bridges. *IEEE Transactions on automation science and engineering* **2014**, *13*, 591-599.
83. Yang, M.-D.; Su, T.-C. Automated diagnosis of sewer pipe defects based on machine learning approaches. *Expert Systems with Applications* **2008**, *35*, 1327-1337.
84. Varadharajan, S.; Jose, S.; Sharma, K.; Wander, L.; Mertz, C. Vision for road inspection. In Proceedings of the IEEE winter conference on applications of computer vision, 2014; pp. 115-122.
85. Li, L.; Sun, L.; Ning, G.; Tan, S. Automatic pavement crack recognition based on BP neural network. *PROMET-Traffic&Transportation* **2014**, *26*, 11-22.
86. Moon, H.-G.; Kim, J.-H. Intelligent crack detecting algorithm on the concrete crack image using neural network. *Proceedings of the 28th ISARC* **2011**, *2011*, 1461-1467.
87. Zhang, A.; Wang, K.C.; Li, B.; Yang, E.; Dai, X.; Peng, Y.; Fei, Y.; Liu, Y.; Li, J.Q.; Chen, C. Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network. *Computer-Aided Civil and Infrastructure Engineering* **2017**, *32*, 805-819.

88. Gopalakrishnan, K.; Khaitan, S.K.; Choudhary, A.; Agrawal, A. Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection. *Construction and building materials* **2017**, *157*, 322-330.
89. Li, H.; Song, D.; Liu, Y.; Li, B. Automatic pavement crack detection by multi-scale image fusion. *IEEE Transactions on Intelligent Transportation Systems* **2018**, *20*, 2025-2036.
90. Tong, Z.; Gao, J.; Han, Z.; Wang, Z. Recognition of asphalt pavement crack length using deep convolutional neural networks. *Road Materials and Pavement Design* **2018**, *19*, 1334-1349.
91. Fan, Z.; Wu, Y.; Lu, J.; Li, W. Automatic pavement crack detection based on structured prediction with the convolutional neural network. *arXiv preprint arXiv:1802.02208* **2018**.
92. Nguyen, N.T.H.; Le, T.H.; Perry, S.; Nguyen, T.T. Pavement crack detection using convolutional neural network. In Proceedings of the Proceedings of the Ninth International Symposium on Information and Communication Technology, 2018; pp. 251-256.
93. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13, 2014; pp. 740-755.
94. Lee, D.; Kim, J.; Lee, D. Robust concrete crack detection using deep learning-based semantic segmentation. *International Journal of Aeronautical and Space Sciences* **2019**, *20*, 287-299.
95. Li, B.; Wang, K.C.; Zhang, A.; Yang, E.; Wang, G. Automatic classification of pavement crack using deep convolutional neural network. *International Journal of Pavement Engineering* **2020**, *21*, 457-463.
96. Fan, Z.; Li, C.; Chen, Y.; Di Mascio, P.; Chen, X.; Zhu, G.; Loprencipe, G. Ensemble of deep convolutional neural networks for automatic pavement crack detection and measurement. *Coatings* **2020**, *10*, 152.
97. Fan, Z.; Li, C.; Chen, Y.; Wei, J.; Loprencipe, G.; Chen, X.; Di Mascio, P. Automatic crack detection on road pavements using encoder-decoder architecture. *Materials* **2020**, *13*, 2960.
98. Guo, J.-M.; Markoni, H.; Lee, J.-D. BARNet: Boundary aware refinement network for crack detection. *IEEE Transactions on Intelligent Transportation Systems* **2021**, *23*, 7343-7358.

99. Liu, H.; Miao, X.; Mertz, C.; Xu, C.; Kong, H. Crackformer: Transformer network for fine-grained crack detection. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021; pp. 3783-3792.

100. Yu, G.; Dong, J.; Wang, Y.; Zhou, X. RUC-Net: A Residual-Unet-Based Convolutional Neural Network for Pixel-Level Pavement Crack Segmentation. *Sensors* **2022**, *23*, 53.

101. Yang, F.; Zhang, L.; Yu, S.; Prokhorov, D.; Mei, X.; Ling, H. Feature pyramid and hierarchical boosting network for pavement crack detection. *IEEE Transactions on Intelligent Transportation Systems* **2019**, *21*, 1525-1535.

102. Lau, S.L.; Chong, E.K.; Yang, X.; Wang, X. Automated pavement crack segmentation using u-net-based convolutional neural network. *IEEE Access* **2020**, *8*, 114892-114899.

103. Shorten, C.; Khoshgoftaar, T.M. A survey on image data augmentation for deep learning. *Journal of big data* **2019**, *6*, 1-48.

104. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**.

105. Lipton, Z.C.; Elkan, C.; Naryanaswamy, B. Optimal thresholding of classifiers to maximize F1 measure. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2014; pp. 225-239.

106. Tsai, Y.-C.; Chatterjee, A. Comprehensive, quantitative crack detection algorithm performance evaluation system. *Journal of Computing in Civil Engineering* **2017**, *31*, 04017047.

107. D6433-18; Standard Practice for Roads and Parking Lots Pavement Condition Index Surveys. ASTM International: West Conshohocken, P., USA, 2018.

108. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* **2011**, *12*, 2825-2830.

109. Bholowalia, P.; Kumar, A. EBK-means: A clustering technique based on elbow method and k-means in WSN. *International Journal of Computer Applications* **2014**, *105*.

110. Barba, S.; Barbarella, M.; Di Benedetto, A.; Fiani, M.; Gujski, L.; Limongiello, M. Accuracy assessment of 3D photogrammetric models from an unmanned aerial vehicle. *Drones* **2019**, *3*, 79.

111. Gujski, L.; di Filippo, A.; Limongiello, M. MACHINE LEARNING CLUSTERING FOR POINT CLOUDS OPTIMISATION VIA FEATURE ANALYSIS IN CULTURAL HERITAGE. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* **2022**.

112. Farella, E.M.; Torresani, A.; Remondino, F. Refining the Joint 3D Processing of Terrestrial and UAV Images Using Quality Measures. *Remote Sensing* **2020**, *12*, 2873.

113. Agisoft, L. Agisoft Metashape user manual. *Agisoft Metashape, September* **2020**, *160*.
114. James, M.R.; Robson, S.; d'Oleire-Oltmanns, S.; Niethammer, U. Optimising UAV topographic surveys processed with structure-from-motion: Ground control quality, quantity and bundle adjustment. *Geomorphology* **2017**, *280*, 51-66.
115. Hartley, R.; Zisserman, A. *Multiple view geometry in computer vision*; Cambridge university press: 2003.
116. Kraus, K. *Photogrammetry: geometry from images and laser scans*; Walter de Gruyter: 2011.
117. Oliphant, T.E. Python for scientific computing. *Computing in science & engineering* **2007**, *9*, 10-20.
118. McKinney, W. Data structures for statistical computing in python. In Proceedings of the Proceedings of the 9th Python in Science Conference, 2010; pp. 51-56.
119. Mirko, S.; Eufemia, T.; Alessandro, R.; Giuseppe, F.; Umberto, F. Assessing the Impact of the Number of GCPS on the Accuracy of Photogrammetric Mapping from UAV Imagery. *Baltic Surveying* **2019**, 43.
120. MATLAB. bwmorph function. Available online: https://www.mathworks.com/help/images/ref/bwmorph.html (accessed on 16/04/2023).
121. MATLAB. bwdist function. Available online: https://www.mathworks.com/help/images/ref/bwdist.html (accessed on 16/04/2023).
122. Bradski, G.; Kaehler, A. *Learning OpenCV: Computer vision with the OpenCV library*; " O'Reilly Media, Inc.": 2008.
123. Agisoft, L. Agisoft PhotoScan user manual. *Professional edition, version* **2016**, *1*, 37.
124. Technology, D.I.M.o.t.F.H.A.R.a. Available online: https://www.fhwa.dot.gov/publications/research/infrastructure/pavements/ltpp/13092/001.cfm (accessed on 16/04/2023).