الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research

جامعة غرداية
University of Ghardaia

N° d'enregistrement
/…../…../…../…../…

كلية العلوم والتكنولوجيا
Faculty of Science and Technology

قسم الرياضيات و الإعلام الآلي
Department of Mathematics and Computer Science

Final thesis, submitted for the purpose of obtaining the degree

Master Thesis

Domain: Mathematics and Computer Science
Field: Computer Science

Specialty: Intelligent Systems for Knowledge Extraction

Theme

Deep Learning-based Surface Crack Detection

Presented by:

Lamine Mohammed Lamine & Nedjar Salah eddine

Publicly defended on July 02, 2025

Before the jury composed of:

Mr. Slimane Bellaouar	MCA	Univ. Ghardaia	President
Ms. Asma Bouchekouf	MAA	Univ. Ghardaia	Examiner
Ms. Nacera Brahim	MAA	Univ. Ghardaia	Supervisor

Academic Year 2024 /2025

Acknowledgments

Praise first to God, who is the root of all success in our lives.

Our sincere thanks go to our supervisor, Mrs. Nacera, who entrusted us with this subject and who took charge of supervising our project, and the interest she showed in our work, her benevolence, her scientific rigour, her fruitful discussions and her high human qualities, were a great help and enabled us to complete this work.

We would also like to express our gratitude to the members of the jury, who did us the honour of agreeing to judge this modest piece of work.

Our sincere thanks go to all our professors in the Computer Science Department at the University of Ghardaia.

Finally, we also extend our thanks to all those who participated with us, from near or far, in completing this project, especially to family and friends.

Dedication

*To **my parents** I owe you what I am today thanks to your love, patience and Countless secrets. May this work be for you a small compensation and appreciation of what you have done so incredibly for me. I lack the words to express all the gratitude, pride, and deep love I have for the sacrifices they have made for my success. May they find here the testimony of my attachment, my appreciation, gratitude, and respect. May God grant them good health and a long life. All my feelings of gratitude to you.*

*To **my sister and brother**, thank you for your moral support and encouragement. I wish you all the happiness and a bright future.*

*To **my dear friends** and all those who provided me with help and support at the right times throughout my studies. I dedicate this work with my deepest gratitude and sincere thanks.*

ML.Lamine

Dedication

To my dear parents with great pleasure, an open heart and indescribable joy, I dedicate my work to my dear, respected and wonderful parents who have supported me throughout my life and whose virtues, sacrifices and human qualities enabled me to live this day. I cannot find the words to express all my gratitude and pride. May the deep love I have for you and the sacrifices you have made for my success find here a testimony of my attachment, gratitude, gratitude and respect. May God keep them healthy, well-being and long life. All my feelings My gratitude to you.

To my sister I hope to live up to your expectations. May this work be the expression of my deep love. Thank you for your moral support and encouragement. I wish you all the happiness you deserve. I wish you a bright future.

To all my family and everyone I love and who loves me.

To my friends To all those who have been able to help and support me at the right times throughout my life My studies. I dedicate this work with gratitude and appreciation.

S.Nedjar

Abstract

Surface crack detection is paramount for ensuring the safety and longevity of civil infrastructure. Traditional manual inspection methods are time-consuming, costly, and operator-dependent. Prior automated methods based on traditional image processing showed promise, though a step forward, but their reliance on handcrafted features and sensitivity to imaging conditions and real-world variations in lighting and texture limited their global applicability. Deep learning is a more powerful tool, but achieving reliable pixel-level remains a significant challenge separation of cracks especially fine, hairline cracks, usually only a few pixels thick with low contrast and easily discarded in the downsampling stages of standard encoder-decoder frameworks; and intricate, or lengthy crack networks. The current paper proposes a better deep learning technique for this task using the UNet++ architecture. This model was chosen specifically over others due to its design strategy. Although standard U-Net innovated vital skip connections to maintain detail, UNet++ enhances that by utilizing nested and dense skip pathways to bridge the semantic gap between shallow encoder and deep decoder. This architecture is especially well-suited to marrying high-resolution spatial data with deep semantic context, which is necessary for being capable of segmenting the fine-grained crack structure accurately. Compared to architectures such as SegNet, which are efficient in terms of computations but lose boundary detail, UNet++ is high-fidelity segmentation optimized and thus is particularly well-suited to this task. The proposed UNet++ model achieved excellent quantitative results: a Dice coefficient of 0.9338, an Intersection over Union (IoU) of 0.8805, an F1-score of 0.9609, precision of 0.9603, and recall of 0.9614, outper forming a baseline U-Net and other contemporary semantic segmentation methods. A prototype system demonstrated the model's applicability for real-world crack de tection tasks. These findings indicate that the UNet++ architecture, coupled with robust data augmentation, offers a highly accurate and reliable solution for pixel level surface crack segmentation, advancing automated structural health monitoring.

Keywords : Deep Learning, Crack Detection, UNet++, Semantic Segmentation

Résumé

La détection des fissures de surface est primordiale pour assurer la sécurité et la longévité des infrastructures civiles. Les méthodes traditionnelles d'inspection manuelle prennent du temps, sont coûteuses et dépendent de l'opérateur. Les méthodes automatisées antérieures basées sur le traitement d'image traditionnel se sont révélées prometteuses, bien qu'un pas en avant, mais leur dépendance à des fonctionnalités artisanales et leur sensibilité aux conditions d'imagerie et aux variations réelles de l'éclairage et de la texture ont limité leur applicabilité globale. L'apprentissage en profondeur est un outil plus puissant, mais la fiabilité au niveau des pixels reste un défi important séparation des fissures particulièrement fines, des fissures capillaires, généralement de seulement quelques pixels d'épaisseur avec un faible contraste et facilement éliminées dans les étapes de sous-échantillonnage des frameworks codeur-décodeur standard; et des réseaux de fissures complexes ou longs. Le présent article propose une meilleure technique d'apprentissage en profondeur pour cette tâche en utilisant l'architecture UNet++. Ce modèle a été choisi spécifiquement par rapport à d'autres en raison de sa stratégie de conception. Bien que standard U-Net ait innové des connexions de saut vitales pour maintenir les détails, UNet++ améliore cela en utilisant des chemins de saut imbriqués et denses pour combler le fossé sémantique entre l'encodeur peu profond et le décodeur profond. Cette architecture est particulièrement bien adaptée pour marier des données spatiales à haute résolution avec un contexte sémantique profond, ce qui est nécessaire pour pouvoir segmenter avec précision la structure de fissure à grain fin. Par rapport aux architectures telles que SegNet, qui sont efficaces en termes de calculs mais perdent des détails sur les limites, UNet++ est une segmentation haute fidélité optimisée et est donc particulièrement bien adaptée à cette tâche. Le modèle UNet++ proposé a obtenu d'excellents résultats quantitatifs: un coefficient de dés de 0,9338, une Intersection sur Union (IoU) de 0,8805, un score F1 de 0,9609, une précision de 0,9603 et un rappel de 0,9614, outper formant une ligne de base U-Net et d'autres méthodes de segmentation sémantique contemporaines. Un système prototype a démontré l'applicabilité du modèle pour des tâches de détection de fissures dans le monde réel. Ces résultats indiquent que l'architecture UNet++, associée à une augmentation robuste des données, offre une solution hautement précise et fiable pour la segmentation des fissures de surface au niveau des pixels, faisant progresser la surveillance automatisée de l'état des structures.

Mots clés : Apprentissage profond, détection de fissures, UNet++, segmentation sémantique

ملخص

يعد اكتشاف الشقوق السطحية أمرًا بالغ الأهمية لضمان سلامة وطول العمر للبنية التحتية المدنية. أساليب التفتيش اليدوي التقليدية تستغرق وقتًا طويلاً ومكلفة وتعتمد على المشغل. أظهرت الأساليب الآلية السابقة القائمة على معالجة الصور التقليدية وعدًا ، على الرغم من أن الخطوة إلى الأمام ، ولكن اعتمادها على الميزات المصنوعة يدويًا وحساسية لظروف التصوير والتغيرات في العالم الحقيقي في الإضاءة والملمس محدودة في قابلية تطبيقها العالمية ، فإن التعلم العميق هو أداة أكثر قوة ، ولكن عادة ما يكون ذلك مجرد تباين منخفض في التباين ، لا يزال هناك أي تحد كبير من التحديات التي تحدها الشقوق بشكل خاص. من أطر عمل التشفير والرائد القياسية ؛ وشبكات الكراك المعقدة ، أو المطولة. تقترح الورقة الحالية تقنية تعليمية عميقة أفضل لهذه المهمة باستخدام بنية UNET++ . تم اختيار هذا النموذج على وجه التحديد على الآخرين بسبب استراتيجية التصميم. على الرغم من أن الاتصالات القياسية المبتكر لـ U-NET للحفاظ على التفاصيل ، إلا أن UNET++ يعزز ذلك من خلال استخدام مسارات تخطي متداخلة وكثيفة لسد الفجوة الدلالية بين التشفير الضحل وفك الترميز العميق. هذه الهندسة المعمارية مناسبة بشكل خاص لدمج البيانات المكانية عالية الدقة مع السياق الدلالي العميق، وهو أمر ضروري للقدرة على تجزئة بنية الشقوق الدقيقة بدقة. بالمقارنة مع البنى مثل SEGNET ، والتي تكون فعالة من حيث الحسابات ولكنها تفقد تفاصيل الحدود ، فإن UNET++ هو تجزئة عالية الحدود مُحسّنة وبالتالي فهي مناسبة بشكل خاص لهذه المهمة. حقق نموذج UNET++ المقترح نتائج كمية ممتازة: معامل النرد البالغ 0.9338 ، تقاطع على الاتحاد (IOU) من 0.8805 ، وهو درجة F1 من 0.9609 ، ودقة 0.9603 ، واستدعاء 0.9614 ، يتفوق على خط الأساس U-NET وغيرها من المقطوعات الدلالية. أظهر نظام النموذج الأولي قابلية تطبيق النموذج على المهام في العالم الحقيقي. تشير هذه النتائج إلى أن بنية UNET++ ، إلى جانب زيادة البيانات القوية ، توفر حلاً دقيقًا وموثوقًا للغاية لتجزئة صدع سطح البكسل ، مما يطور مراقبة الصحة الهيكلية الآلية.

الكلمات المفتاحية : التعلم العميق، كشف التصدعات (الشقوق)، U-Net++ ، التجزئة الدلالية

Contents

List of Figures

List of Tables

List of Acronyms

- **Adam:** Adaptive Moment Estimation.
- **AI:** Artificial Intelligence.
- **AMP:** Automatic Mixed Precision.
- **ANNs:** Artificial Neural Networks.
- **AP:** Average Precision.
- **AR:** Average Recall.
- **CNNs:** Convolutional Neural Networks.
- **DCNNs:** Deep Convolutional Neural Networks.
- **DDLCN:** Deep Dictionary Learning and Encoding Networks.
- **DL:** Deep Learning.
- **FCNs:** Fully Convolutional Networks.
- **FLOPs:** Floating Point Operations Per Second.
- **FP16:** Half Precision (16-bit Floating Point).
- **FPS:** Frames Per Second.
- **GPUs:** Graphics Processing Units.

- **GUI:** Graphical User Interface.
- **IoU:** Intersection over Union.
- **IPTs:** Image Processing Techniques.
- **LBP:** Local Binary Patterns.
- **LSKA:** Large Separable Kernel Attention.
- **ML:** Machine Learning.
- **MVI:** Manual Visual Inspection.
- **RAM:** Random Access Memory.
- **ReLU:** Rectified Linear Unit.
- **YOLO:** You Only Look Once.

Introduction

The structural integrity of civil infrastructure, encompassing bridges, roads, tunnels, and buildings, is paramount for public safety and economic stability. Sur face cracks are the most basic indicators of material degradation and structural distress. The traditional method for detecting cracks is by employing Manual Visual Inspection (MVI). MVI methods for crack detection and other forms of visual inspections can be limited since they are usually time-consuming, labor-intensive, subjective, and typically expose inspectors to hazards. These limitations indicate the need to develop automated, objective, and efficient methods of inspection.

Early efforts were devoted to automating the detection using conventional Image Processing Techniques (IPTs) and conventional Machine Learning (ML). Early practices were rooted in central image processing concepts like the detection of edges, represented by the highly influential Canny edge detector [7], and automatic thresholding to separate crack pixels [8]. They were also developed more advanced systems, applying methods such as the continuous wavelet transform to analyze features on numerous varying scales [9], graph-theory models such as the Cracktree system in order to connect potential crack pieces [10], and even fuzzy logic in order to deal with image uncertainty [11]. In parallel, other studies tended to feed these manually created features to conventional classifiers most importantly Support Vector Machines (SVMs) [12] and shallow early neural networks [13] to automatically classify image regions.

The advent of Deep Learning (DL), particularly Convolutional Neural Networks (CNNs), was a paradigm shift as it enabled the model to learn on its own from raw data and thus avoid the inherent weakness of earlier methods. The problem can be conceptualized in multiple ways within the DL community. Straightforward activities like image categorization or object recognition also possess their own constraints for this specific job. Image classification is able to ensure the presence of a crack in a patch but not localize it, while object detection provides coarse localization with geometrically unsuitable rectangular bounding boxes for describing thin curvilinear crack patterns.

For proper structural assessment, geometric measurement accurately is required. This has made semantic segmentation the labeling of every pixel in a photograph, assigning a class label (e.g., 'crack' or 'background') within the most suitable methodology. This approach provides a high-resolution pixel-wise map that can accurately label a crack's length, width, and topology. The aim of this thesis is therefore to study, implement, and experiment a deep learning-designed semantic segmentation model for crack automatic detection.

The purpose of this thesis is to study, implement, and evaluate a deep learning-based semantic segmentation model for automated crack detection in civil infrastructure. More specifically, this research will evaluate U-Net, and an advanced version of U-Net called U-Net++, since both are well-known for their considerable success in biomedical and other complex segmentation applications. The methodology sections will cover training these models using a publicly available crack dataset, evaluating performance using common metrics, comparing them to existing approaches, and discussing the potential of the models using a simple crack detection system.

The thesis is structured as follows:

Chapter 1: Foundations and Problem Formulation lays the foundational groundwork, starting with the significance of crack detection and the critical limitations of traditional inspection methods. It then charts the evolution toward automated approaches, highlighting how the reliance on hand-crafted features in classical computer vision necessitated the paradigm shift to Deep Learning. The chapter concludes by defining the key challenges inherent to modern DL-based solutions, thereby framing the problem this thesis aims to solve.

Chapter 2: State of the Art and Methodological Context provides a comprehensive review of the state of the art in automated crack detection. It critically analyzes the progression of deep learning applications from coarse image classification and object detection to the currently favored, more precise semantic segmentation approaches. By surveying seminal and contemporary works, this chapter establishes the methodological context and justifies the focus on segmentation as the most effective paradigm for quantitative structural analysis.

Chapter 3 details the experimental setup and implementation, including the dataset used, data preprocessing and augmentation techniques, the proposed model architectures (U-Net and U-Net++), the training process, evaluation metrics,

presentation of results, a comparative analysis with previous methods, and the development of a simple crack detection system, along with a discussion on limitations and future directions.

Finally, a General Conclusion summarizes the key findings, contributions, and potential avenues for future research.

Chapter 1

Generalities and preliminary knowledge

Introduction

This chapter serves as a fundamental overview of surface crack detection with a focus on the shift from Manual Visual Inspection to Deep Learning Automation. The traditional Manual Visual Inspection (MVI) methods are limited by their inherent subjectivity, high cost, and safety challenges, requiring more rigorous and objective methods. In this chapter, we have summarized the history of surface crack detection starting with one component of MVI. It is obviously an advantage over manual inspection, yet the earliest of these systems were absolutely dependent on hand-crafted features from classical Image Processing Techniques (IPTs) and traditional Machine Learning (ML) approaches. This situation resulted in a schema whereby improvements to the overall product were solely dependent on either incorporating quality control measures from the MVI phase or improving Image Processing Techniques (IPT) in conjunction with supervised training of the machine learning models. Thus, the evolution from MVI resulted in the need for Deep Learning (DL) which reduces the need for hand-crafted features by allowing the DL model to extract features from the raw image data. Ultimately, we wanted to provide a description of the technical challenges that still exist in the DL domain with respect to surface crack detection - particularly data annotations costs, class imbalance issues, and fine-grained crack typologies - all of which form the technical framework for the sophisticated segmentation models we operated within this thesis.

1.1 Definition and Significance

Surface cracks are discontinuities or breaks that form on the material's surface, usually in civil engineering structures like bridge decks, concrete pavements, exterior walls of buildings, asphalt roads, tunnels, and rail components [13, 14, 15]. Regardless of whether these are minor at the start or huge, they are highly critical indications of material deterioration and structural distress [16, 14]. They have threefold significance: they impair structural integrity by reducing load-bearing capacity and modifying stress patterns directly [17]; they undermine durability dramatically by creating routes for water, chlorides, and other attacking chemicals to find their way inside and cause or promote internal harm mechanisms like corrosion of reinforcement or freeze-thaw damage [18, 14]; and they ultimately influence safety by potentially triggering secondary deterioration and, in the most serious cases, structural failure [13, 19]. Therefore, monitoring of cracks is significant for maintenance as well as maintaining continuous serviceability and safety of key infrastructure [20, 13, 14].

1.2 General Causes

Cracking results from a series of causes like material aging, mechanical loads stresses (vehicle, operating loads), environmental conditions (temperature fluctuation causing thermal stress, humidity change, chemical attack, UV radiations), shrinkage (plastic, evaporation), and material or construction defects [14, 21].

1.3 The Problem of Crack Detection

1.3.1 Need for Inspection

Material fact of the behavior requires that cracks are essentially unavoidable in civil infrastructure throughout its lifespan [14]. These cracks, as just outlined, are more than mere surface blemishes; they are crucial indicators to underpinning structural health and potential weaknesses [16]. The inherent necessity for routine and organized inspection all stems as a result of the need to control risks pertaining to the inadequacies. Those infrastructures made up of roads, bridges, and buildings are pillars on which today's world stands, and collapsing or breaking may not only cause considerable economic blow and loss, most importantly but potentially lead to death. The inspection is therefore unambiguously imperative for:

Public Safety: This is the most critical one. Early identification of potentially critical cracks before structural failure is necessary in order to protect users and the public [13, 19].

Serviceability: Cracks can compromise the intended performance of a structure (e.g., ride comfort on pavements, water tightness in tunnels or buildings). Inspection exposes issues that affect serviceability, allowing remediation before it is too late.

Integrity and Long-term Service Life: Cracking accelerates degradation by allowing deleterious agents to penetrate the structure [14]. Early detection and restoration (e.g., sealing cracks) can effectively slow this degradation, preserving the asset in a sound condition while extending its effective service life beyond original estimates [20].

Maintenance Resource Optimization: Infrastructure owners manage enormous sets of assets with limited budgets. Condition data collected through inspection allows maintenance and repair work to be prioritized, focusing resources where they are most needed and moving away from costly reactive repairs to less expensive, planned, proactive maintenance [14, 13].

Regulatory Compliance: Regular compulsory structural inspections are required by law in the majority of jurisdictions to meet safety requirements [22, 21].

1.3.2 Traditional Inspection Methods (Manual Visual Inspection)

The prevalent method over the past decades for addressing crack inspection demand has been Manual Visual Inspection (MVI) [21, 23] as described in Figure 1.1. Through this method, inspectors as humans, in general, engineers or experienced technicians, personally travel to the structure and examine its surface meticulously [16].

Process: Visual inspection is the primary task. Inspectors visually inspect surfaces, sometimes along predetermined paths or across known critical areas, for any discontinuities that indicate cracking [14]. Simple hand tools can be used, such as magnifying lenses to examine fine details, crack width gauges or comparators

Figure 1.1: Crack detection by manual visual inspection (MVI).

for basic measurements, marking tools (paint and chalk) to define boundaries, and cameras to document images [21]. For concrete, acoustic methods such as hammer sounding, described in Figure 1.2, can be additionally used to detect scaling that often occurs with cracking.

Access: The second major element of MVI is reaching the involved surfaces, which may include specialized equipment like scaffolding, cherry pickers, man-lifts, UBIUs, or even rope access systems in the scenario of bridges and structures [21]. This usually involves closing traffic or service disruption.

Recording: Observations are generally recorded by hand on field sheets, standard forms, or sketches, usually supplemented by photographs. Formal inspection reports are then formalized by summarizing the field notes [16].

Figure 1.2: Hammer sounding and chain-drag testing [1].

1.3.3 Limitations of Traditional Inspection Methods

Albeit being antiquated and continuing to be utilized, MVI has inherent weaknesses that significantly influence its validity, effectiveness, cost-effectiveness, and applicability to current infrastructure management [14, 16, 21, 24]:

There is a Lack of Consistency and Subjectivity: It is one of the greatest flaws. Crack detection and assessment heavily depend on the individual inspector's vision, training, experience, alertness, current fatigue level, and even individual biases [21, 16, 25, 26]. Definitions of crack severity (e.g., hairline vs. fine vs. medium) can vary, and the identification of small cracks heavily depends on the observer. This leads to very large inconsistency of results across various inspectors (low reproducibility) and even across the same inspector at different times (low repeatability). This inconsistency makes it extremely difficult to reliably compare condition states over time or across different structures [16].

Time-Consuming and Labor-Intensive: MVI entails highly skilled staff taking significant time to travel, setup (access equipment, traffic control), meticulous surface scanning, and documentation [14, 21, 15]. Scanning large or complex structures can take days, weeks, or months, consuming significant human resources.

High Costs: The extensive labor hours, together with the direct and indirect cost of hire equipment access, traffic control, potential revenue loss due to interruption of services, travel, and report production, make MVI an expensive undertaking [21, 16, 14].

Safety Risks: Inspectors have much of their working lives spent laboring under hazardous conditions—at height, in confined spaces, near live traffic, exposed to weather conditions, or on structures potentially weakened—presenting significant

safety hazards [21, 27, 15]. Reducing human exposure to these conditions is a primary goal of automation.

Scalability and Coverage Problems: The cost and time requirements are great, hence MVI cannot scale up very conveniently to the massive amounts of existing infrastructure. Often enough, inspections are not done at the desired level or frequency due to budget and resource constraints [16]. Also, physical access limitations impede adequate inspection of certain structural components [21].

Limited Detection Capabilities: Human vision, specifically in the field environment (varying lighting, distance, glancing angles of view), is unable to consistently see very thin hairline cracks or cracks obscured by surface soiling (dirt, efflorescence) or complex textures [28]. The initial damage may go unnoticed until it becomes extensive.

Data Format and Administration: Manually recorded data (notes, drawings, photos) can be difficult to integrate into digital asset management systems, and quantitative analysis, trend monitoring, and decision-making based on data can be less effective than natively digital automated outcomes [16].

A cumulative effect of such deep flaws of traditional manual visual inspection is the compelling necessity for new solutions. Calls for greater objectivity, consistency, affordability, cost-savings, security, and expandability in getting detailed, quantifiable information have been the foremost driver of R&D into computerized crack detection technology, first based on IPTs and traditional ML, now increasingly driven by advances in Deep Learning.

1.4 Pre-Deep Learning Automated Detection Approaches

Driven by the need for surpassing the limitations of conventional visual inspection (subjectivity, cost, time, safety, scalability [21, 16, 26]), significant quantities of research effort went into development of computer vision-driven automatic surface crack detection techniques before the popularization of deep learning. These methods can be broadly classified as those that are primarily Image Processing Technique (IPT) based and those which integrate IPTs with traditional Machine Learning (ML).

Figure 1.3: A close-up image of a crack in a concrete surface.

1.4.1 Image Definition

The image is the original input photo of a material surface (e.g., concrete, asphalt, or metal) taken by a camera. It shows the real-world scene, possibly containing visible cracks. These images serve as the input data for the crack detection model.

Format: RGB as shown in Figure 1.3 or grayscale image.

Content: Surface with or without cracks.

1.4.2 Mask Definition

The mask (also called a ground truth mask or label) is a binary or multi-class image of the same size as the original image. It indicates the locations of cracks in a pixel-wise manner.

Format: Usually a binary image (same height and width as the original image) as shown in Figure 1.4.

- 1 (white) = crack
- 0 (black) = background (non-crack)

1.4.3 Image Processing Techniques (IPTs)

This algorithm family spans a wide range of algorithms aimed at inspecting digital images to extract some information, here crack features. The general idea was typically to leverage observed intensity, shape, or texture contrast between background and pixels in the crack, for example. Typical IPT families used for

Figure 1.4: The binary ground truth mask for the original image.

crack detection are:

Thresholding Methods: Of the easier techniques, those which attempt to bypass segmentation by breaking up the image's histogram based on intensity of pixels. The general assumption made here is that cracks have a tendency to be darker (or maybe lighter, depending upon condition and image type) compared to the background [29]. Even though basic global thresholding does not work typically in uneven illumination and low contrast, newer methods came in its place, e.g., adaptive thresholding (thresholds are calculated locally), Otsu's thresholding (maximizing intra-class variance) [8], valley-emphasis methods [13], or even fuzzy logic-based thresholding in order to address uncertainty [11]. But thresholding alone will result only in noisy and fragmented responses.

Edge Detection: Cracks will likely be represented as sharp changes or edges in image intensity. Cracks will likely be detected by edge detection algorithms. Primarily utilized are Sobel and Prewitt operators (approximation of gradient through simple convolution) [30, 23] and more sophisticated Canny edge detector (with hysteresis thresholding and non-maximum suppression for better edge linking) [30, 28, 7].An example is shown in Figure 1.5. Though proficient at detecting edges, they are image-noise sensitive and will identify plenty of spurious non-crack edges (e.g., aggregate boundary, texture, joint edges in pavements), and thereby result in extremely high false positive rates and tons of subsequent post-processing [23].

Morphological Operations: These procedures involve the use of structuring elements (pre-defined small objects) to examine the image's geometry. For crack detection, they are utilized in subsequent post-processing operations after preliminary

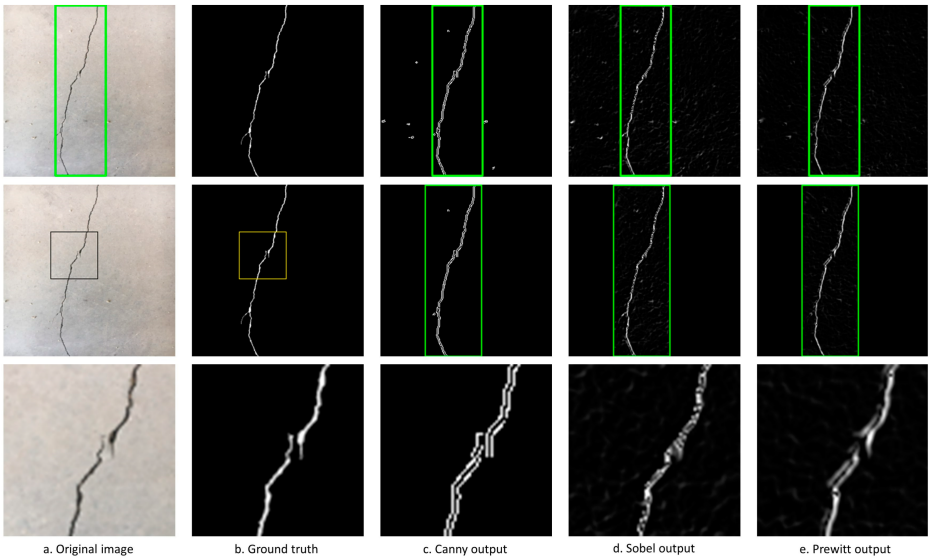

Figure 1.5: Visual comparison of edge detection results [2].

detection. Morphological closing (dilation of erosion) will close up small detected cracks openings or bridge close proximity fragments [23]. Morphological opening (erosion followed by dilation) will remove small noise objects with a restricted size. Thinning or skeletonization algorithms convert recognized crack areas to centerlines of one pixel width, which is easier to measure length and topology analysis subsequently [31, 23].

Filtering: Filters are applied for feature enhancement or noise removal. Median filters are commonly employed in noise removal since they are capable of removing salt-and-pepper noise and have fewer chances of blurring edges compared to linear filters like Gaussian filters [23, 29]. More advanced filtering techniques specifically studied for cracks are anisotropic diffusion (in the direction of edge-preserving smoothing) [29], wavelet transforms (to analyze features at different scales and frequency, possibly separating cracks from noise) [9], Gabor filters (to extract crack-related information in texture and orientation) [32], and Bidimensional Empirical Mode Decomposition (BEMD) (a data-driven algorithm to decompose images into modes, possibly discriminating noise or crack features) [30].

Texture Analysis: These methods examine the spatial texture pattern of pixel values and, on the assumption that cracks have a different contrasting texture pattern than the remainder of the background, detect cracks by having a unique different texture pattern. Examples include the use of Local Binary Patterns (LBP) or GLCM feature-based descriptors known as Haralick features [33]. These texture

characteristics themselves were not generally used directly for the detection but as inputs to subsequent ML classifiers.

Graph-Based and Path-Finding Techniques: Some techniques mapped out potential crack pixels or segments as graph vertices and utilized graph traversal methods (e.g., minimum path finding, Minimum Spanning Tree generation) to connect fragments into reasonable crack lines, often based on perceptual features such as proximity and continuity [31, 10].

1.4.4 Limitations of Traditional IPTs

Despite the variability of the methodology, IPT-based methodologies as a whole were not that flexible and robust. They were highly sensitive to certain image conditions (light, view, presence or absence of shadowing, noise level, blurriness) [30, 28]. They were most likely to require careful, by-hand parameter adjustment on a case-by-case or per-data-set basis [22]. Distinguishing actual cracks from fine background texture details, stains, joints, filled cracks, or other fractured-like features was a challenging process, tending to cause over-occurrence of false positive or fractured detection [23, 10]. They essentially relied on assumptions about what cracks look like (e.g., lower intensity, edge occurrence), which may not hold everywhere.

1.4.5 Conventional Machine Learning Approaches (ML)

To improve decision-making capabilities and possibly introduce robustness, researchers have combined IPTs with conventional ML classifiers. This was a deviation from purely rule-based systems toward data-driven yet still feature-explicit approaches.

Two-Stage Process: The standard pipeline included:

1. **Handcrafted Feature Extraction:** Engineers would handcraft a feature set thought to be beneficial for crack differentiation by combining some of the IPTs mentioned above (thresholding output, edge responses, texture features, shape descriptors, pixel intensity statistical summaries in regions, etc.) [13, 33]. This feature engineering process was still necessary and required a lot of domain knowledge and trial-and-error.

2. **Classifier Training:** The extracted features, typically arranged into a feature vector per image patch or region, were then used to train a standard ML classifier. Traditional choices were Support Vector Machines (SVM) [12, 33, 32], traditional Artificial Neural Networks (ANNs, i.e., multi-layer perceptrons) [13, 34, 32], Random Forests (RF) [16], or Boosting algorithms [32]. These models performed a mapping from the handcrafted feature space to the class labels (crack/non-crack)..

Main Shortcoming: Although generally improving class accuracy compared to purely IPT-based rules, the inherent shortcoming of conventional ML approaches to breaking identification was their dependence on engineered features [28, 33, 16]. The system's overall performance was subsequently limited by the discriminability and quality of the handcrafted features defined by the human designer. If the chosen features did not control for the intrinsic variations or were non-intrinsic variable-sensitive, even the best classifier would fail. This limitation in feature engineering prohibited the development of truly robust and general automatic crack detection systems that could operate in a range of real-world conditions, and this led to the breakthrough in deep learning.

1.5 Deep Learning (DL)

The constraints of the traditional Image Processing Techniques (IPTs) and classical Machine Learning (ML) approaches, particularly their reliance on hand-crafted features and susceptibility to real-world fluctuations, necessitated a shift towards more resilient and adaptive solutions for computer vision inspection tasks like crack detection. This transformation was initiated by the advent of Deep Learning (DL), one of the streams of machine learning that has transformed computer vision and numerous other domains of Artificial Intelligence [35, 36].

Definition

Deep Learning employs Artificial Neural Networks (ANNs) of multiple layers (and hence "deep") situated between input and output. Roughly inspired by hierarchical processing believed to occur within living nervous systems, the layers progressively reinterpret the input data into progressively higher levels of abstraction and complexity [36]. Unlike conventional ML in which feature extraction is a direct, standalone process, DL aims to directly learn intricate patterns and hierarchical features from the raw data by a series of non-linear transformations

between these layers. The key notion is to allow the model itself discover the most salient features for a given task by minimizing its vast number of internal parameters (weights and biases) during training on large datasets.

1.6 Key Challenges in DL-Based Crack Detection and Segmentation

In spite of the unprecedented achievement accomplished with Deep Learning (DL) for surface crack detection automation, several challenges remain relevant to restraining progress in making well-rounded, universally applicable, and easily deployable systems. These challenges emanate from the nature of cracks, natural-world imaging conditions' imperfections, DL model inherent requirements, and implementational deployment considerations.

1.6.1 Data Sparsity and Annotation Cost

The Need for Gigantic Datasets

DL models, especially deep structures like CNNs and Transformers, are notoriously known to be hungry for data. They typically require gigantic volumes of labeled data in order to learn generalizable features and avoid overfitting [28, 16, 14]. While big public datasets of computer vision overall (e.g., ImageNet) do exist, domain-specialized datasets for infrastructure cracks are usually relatively small [37, 16, 38].

Annotation Bottleneck (Especially for Segmentation)

The major bottleneck is collecting sufficient labeled data. While image/patch-level classification labels (crack/non-crack) are simple to obtain, generating precise pixel-level segmentation masks is extremely time-consuming, labor-intensive, and requires a lot of domain knowledge [39, 16, 40]. This manual process limits the size and variety of publicly segmented datasets [37, 29], making it hard to train generalized models over many structure types, materials, and environmental conditions. Subjectivity of manual labeling, especially at crack boundaries, introduces label noise as well [16].

Mitigation Strategies

Studies examine data augmentation [41, 15], transfer learning (fine-tuning pre-trained models on big datasets like ImageNet or COCO) [42, 16, 14], semi-supervised learning (using lots of unlabeled data in combination with limited labeled data) [37, 40], weakly supervised learning, and synthetic data generation [37] to reduce dependence on large fully annotated datasets.

1.6.2 Class Imbalance

Problem Definition

In regular surface images, crack pixels are just an extremely small percentage (typically «1%) of the total pixels compared to the background [22, 39, 24, 38]. Such tremendous disproportion between foreground (crack) and background classes is a critical problem to be encountered during the training process.

Impact

Standard loss functions (e.g., pixel-wise cross-entropy) can be overwhelmed by the easily labeled background pixels. The model may achieve high overall accuracy by labeling everything as background, failing to learn useful representations for the minority crack class, and hence having poor crack detection performance (low recall) [24, 38].

Mitigation Techniques

Techniques to fight this are employing tailored loss functions that put higher weight on the minority class or focus on challenging examples (e.g., weighted cross-entropy, Focal Loss [43], Dice Loss [39], IoU Loss [38], hybrid losses [43]), specific sampling methods at training time (e.g., oversampling crack pixels/patches [44]), or structural modifications.

1.6.3 Detection of Fine, Faint, and Complex Cracks

Subtle Features

Cracks can be extremely thin (hairline cracks, perhaps only 1-2 pixels wide), weak (low contrast with the background), or partially occluded [22, 39, 24]. Such

subtle features can easily get lost in the downsampling steps in typical encoder-decoder architectures or completely ignored if their intensity profile is not adequately distinct.

Complex Patterns

Cracks are intricate patterns (e.g., alligator cracking) with many branches and intersections, or occur with greatly irregular paths [10]. It is challenging for models with appropriate receptive fields to learn the whole topology and connectivity of complex patterns. Simple CNNs with limited receptive fields might not be capable of handling that.

Mitigation Strategies

Architectures that maintain high-resolution detail (e.g., U-Net skip connections [16]), avoid aggressive downsampling [16], use multi-scale feature fusion [22, 45], or implement attention mechanisms/transformers so as to get long-range context [39, 15] are likely to be beneficial.

1.6.4 Robustness to Real-World Variations

Environmental Conditions

Field observation images are subject to very irregular conditions like uneven and problematic illumination (sunny areas, shadows, dark spots) [28, 10], surface conditions (dirt, stains, water, uneven textures, different materials) [22, 29], and the presence of noise or blur due to sensor deficits or motion artifacts [28, 41].

Crack-Like Characteristics

It is important to identify real cracks from other linear features or surface blemishes (e.g., pavement joints, filled cracks, scratches, utility marks, object boundaries, rust stains, tire marks) to minimize false positives [22, 16, 23].

Mitigation Strategies

Training on different datasets that explicitly represent these variations is important for resilience [22]. Data augmentation techniques simulating such conditions are standard practice [41, 15]. Architectures with the ability to learn invariant features are preferred.

1.6.5 Computational Efficiency and Deployment

Real-Time Processing

For most uses, especially persistent observation from vehicles in motion (drones), ongoing or virtual ongoing processing within real-time is necessary [22]. Nevertheless, advanced DL segmentation models might be computationally intensive.

Lightweight Models

Deployment of models on resource-constrained hardware like mobile phones, embedded systems of UAVs, or edge computing platforms needs lighter models with fewer parameters and lower FLOPs at the expense of sacrificing little on accuracy [22, 46, 41, 43, 47].

Mitigation Strategies

Research focuses on efficient architecture design (e.g., the use of depth-wise separable convolutions, group convolutions, specific modules like GhostNet or GSConv [22, 46, 47]), model compression techniques (pruning, quantization), and hardware acceleration.

Conclusion

In this chapter, we have identified the scientific and practical rationale for utilizing deep learning in crack detection. We demonstrated the scientific limitations of human inspection (which can be described as both subjective and inefficient), and compared those pitfalls to the shortcomings of earlier work that sought to automate visual inspection through an explicit feature engineering phase. We demonstrated a clear case for a paradigm shift toward end-to-end learning. The main takeaway is, while Deep Learning can solve the issue of feature design, the application of Deep

Learning introduces a new set of specific problems to contend with - including data sparsity, extreme class imbalance (which we will see in Chapter 2), and the need for high computational efficiency. The results of that analysis set the stage for a review of specific DL architectures in Chapter 2, where we outline what the solutions were to the problems discussed here.

Chapter 2

Related Work

Introduction

Based on the shared challenges presented in Chapter 1, this chapter discusses the particular application of deep learning methods in detecting cracks. It begins by framing the problem in the three broad computer vision paradigms of image classification, object detection, and semantic segmentation. One of the primary stress points lies in describing why semantic segmentation stands out, with explication of its sole facility for precise geometric quantification and morphological analysis that is of utmost importance for structure assessment. The chapter continues with a formal summary of the most significant architectures enabling pixel-level examination, including specialization within encoder-decoder models from the original U-Net to more advanced variants like UNet++. Finally, a comprehensive review of the seminal and current literature informs these principles, surveying the manner in which these tasks and structures have been achieved and advanced by the academic community to this purpose.

2.1 Foundations of Deep Learning for Crack Detection

Deep Learning (DL) is distinguished from traditional Machine Learning (ML) primarily through automatic feature learning, or representation learning [28, 35], whereby models acquire sophisticated and robust discriminative features indirectly from data during training [16, 14], in place of hand-designed features and end-to-end learning from raw pixels to final output being natively unsupported [16]. The foundation of this computer vision methodology is the Convolutional Neural

Network (CNN), the de facto standard for visual tasks including crack detection [16, 48, 49], constructed from specialized building blocks—like convolution layers with parameter sharing to identify translation-invariant features [35], pooling layers to down-sample and conserve memory [22, 35], and non-linear activation functions [28, 14] to develop a hierarchical representation of visual information, learning low-level edges to high-level, task-specific patterns [22, 36]. The training of these models follows several paradigms, the most common one being supervised learning [37, 16], in which a model learns a mapping from inputs directly to ground-truth labels as they are familiar [35]; while this can be extremely accurate, its reliance heavily on large quantities of expensive, time-consuming labeled data places a massive bottleneck, especially for pixel-level segmentation [16, 39, 40]. To respond to this, unsupervised learning is done without any explicit labels [35], typically by training on typical, crack-free surfaces in order to detect cracks as abnormalities [37], though this can struggle with the accuracy and vast variety of normal surface appearances. Placing itself midway between these two poles, semi-supervised learning (SSL) offers a reasonable compromise by leveraging on a small amount of labeled data alongside enormous amounts of unlabeled data [40], by employing consistency regularization and pseudo-labeling techniques to achieve performance levels equivalent to fully supervised methods at significantly reduced annotation costs [16], thus making high-performance DL solutions more affordable and cost-effective for deployment on real-world infrastructure inspection [37, 16, 40].

2.2 Deep Learning Tasks for Crack Detection

The successful implementation of deep learning for crack detection is dependent on the formalization of the inspection objective in the form of an explicit computer vision paradigm. This problem formulation is crucial because it characterizes the overall process and the nature of the output, ranging from a simple binary classification to a dense pixel-wise map. The paradigms that were most researched for this purpose Image/Patch Classification, Object Detection, and Semantic Segmentation are each an optimal compromise between the granularity of spatial information, associated annotation effort, and model computational demand. Selecting the best task is therefore not at random but depends on the specific engineering requirements, whether this is initial screening, coarse localization, or precise geometric and morphological measurement of defects.

2.2.1 Framing the Problem

After the capability of Deep Learning (DL) to learn automatically from visual data was achieved, the subsequent challenge was to position the specific problem of crack detection within the realm of established computer vision task paradigms. The choice of framework decides the type of output generated, the type of training data annotations required, the suitable network architectures, and lastly, the level of detail provided about the found cracks. Depending on the specific goals of the inspection—ranging from simple detection of damage presence to precise mapping and quantification thereof—crack detection using DL is largely dealt with by three main tasks: Image/Patch Classification, Object Detection, and Semantic Segmentation.

2.2.2 Image/Patch Classification

Patch/image classification would be the simplest formulation of crack detection, which would be a single-label classification problem such that one would predict an input image unit to be one of the class labels, for instance, "crack" or "non-crack" [28, 42]. Since large-scale infrastructure images are usually of high intra-image variability and whole-image labeling becomes not feasible, the approach is most effective at the patch level, where the surface image is divided into small, preferably square patches e.g., patch sizes of 99×99 [32] or 256×256 [28, 42] have been used and each patch is processed separately. A generic CNN classification network, ranging from shallow handcrafted networks to well-known backbones like AlexNet [14], VGG [42], or ResNet [21], is learned on labeled patches. In the case of large inspection images, a sliding window mechanism is commonly used, scanning patches in a dense manner over the image, classifying each one with the learned CNN, and combining the results into an approximate probability map of cracked regions [28, 22]. Annotation needs are minimal, with patch-level or image-level labels (crack/non-crack) being sufficient, much less time-consuming compared to pixel-level annotation. With conceptual simplicity, minimal annotation effort, and effective utilization of transfer learning from pre-trained networks on big datasets like ImageNet very useful where labeled crack data is scarce [21, 14, 42], it is optimally utilized for preliminary screening or filtering of images that are likely to contain damage. But it provides only rough localization, that is, a patch is cracked without the indication of where its center is or the width of it. The sliding window algorithm is computationally costly as it contains repeated processing of overlapping pixels [22], and the resulting crack maps are blocky, resolution-dependent, and unsuitable for direct measurement-based geometry like length or width.

2.2.3 Object Detection

Object detection aims to locate and categorize cracks within an image and report a list of bounding boxes (typically rectangles) around the detected cracks, together with a class label (e.g., "crack") and a confidence score [16]. Architectures for this task generally follow standard object detection architectures, divided into two-stage detectors such as the Faster R-CNN family [16, 50], which first propose regions and then classify them, and one-stage detectors such as the YOLO family [51, 52] and SSD citeliu2016ssd, which perform localization and classification simultaneously. Annotation involves drawing bounding boxes around all crack instances in the training images. Object detection gives direct localization feedback and is more inference-efficient, at least for one-stage detectors, compared to sliding window patch classification. Its main limitation for crack analysis is the low geometric appropriateness of rectangular bounding boxes for representing elongated, curvilinear, and often networked thin cracks [16, 15]. Bounding boxes may enclose large regions of background or only partially encompass lengthy crack systems, and it is difficult to glean morphological information such as precise width or length. Hence, object detection is seldom the preferred option for detailed crack analysis over segmentation.

2.2.4 Semantic Segmentation

Semantic segmentation provides the most accurate type of analysis by performing pixel-level labeling [16], a class label (for instance, "crack" or "background") to every pixel of the input image. It outputs a segmentation mask, an output normally a binary image the same size as the input, where crack pixels are marked [22]. This is mainly achieved by Fully Convolutional Networks (FCNs) [53], which replace the dense classification layers in CNNs with convolutional layers for maintaining spatial information. Most current segmentation networks are an encoder–decoder architecture [39, 40, 47], where the encoder (for example, VGG, ResNet, or EfficientNet) progressively downsamples to learn high-level semantic data and the decoder progressively upsamples, sometimes injecting information from previous encoder layers via skip connections, as initially developed in U-Net [54], or using specialized upsampling layers, as in SegNet [4]. Performance has been improved subsequently by applying higher-order abstractions such as atrous convolutions (DeepLab family) [55], dense connectivity [24], attention mechanisms [15, 43], and transformers [41, 39]. The most computationally expensive step in the annotation process for semantic segmentation is the process of annotation, which includes pixel-to-pixel ground truth masks in which each crack pixel is hand-annotated [16, 40]. But this method provides the highest localization precision, distinctly

describing spatial position and geometry of cracks, and enabling direct measurement of the key geometric parameters such as length (skeletal orientation), alternating width, area, density, and directionality [23, 29, 47]. It is the best approach to thorough condition inspection, deterioration assessment, and accurate repair planning, and has left a permanent impression on standard deep learning techniques in crack detection [37, 16], although requiring high annotation effort and potentially more computationally demanding than classification—issues addressed in successful applications [22, 43].

These three deep learning paradigms – Image/Patch Classification, Object Detection, and Semantic Segmentation – offer varying means of framing the crack detection problem, each with its own relative strengths and weaknesses regarding output resolution, annotation requirements, and suitability for analysis. The most notable differences are presented in Table 2.1.

Table 2.1: Comparison of Deep Learning task formulations for surface crack detection.

Feature/Aspect	**Image/Patch Classification**	**Object Detection**	**Semantic Segmentation**
Primary Output	Single Class Label (Crack / No-Crack) per image/patch.	Bounding Boxes around detected cracks + Class Label + Score.	Pixel-level Classification Map (Segmentation Mask).
Level of Detail	**Coarse:** Indicates presence/absence within the input unit.	**Moderate:** Locates a rectangular region containing the crack.	**Fine-grained:** Precisely outlines the crack's shape at the pixel level.
Localization Accuracy	Low (Limited to the patch/image boundary).	Moderate (Limited by the bounding box shape).	High (Pixel-level precision).
Annotation Required	Image/Patch-level Labels (Least effort/cost).	Bounding Box Coordinates for each crack (Moderate effort).	Pixel-level Masks for all crack pixels (Highest effort/cost).
Handling Crack Geometry	Does not capture geometry.	Poor fit for thin, long, branching, or networked cracks (rectangular boxes).	Excellent fit, directly maps the crack's true shape and continuity.
Typical Use Case	Initial screening, filtering data, simple presence detection.	Locating distinct crack regions, counting separate major cracks.	Detailed analysis, precise mapping, geometric quantification, severity assessment, monitoring propagation.

2.3 Motivation for DL-Based Crack Segmentation

While DL-based image classification and object detection offer helpful functionality for coarse localization and first-level screening of cracks [28, 52], they fall short in providing the rich, fine-grained information required for thorough structural assessment and well-informed maintenance decisions. The inherent weaknesses of such approaches—i.e., the coarse localization provided by patch classification [22] and the geometrically poor fit of bounding boxes to intricate crack patterns [16, 15]—mandate a more detailed-grained approach. This need for precise spatial understanding is the underlying cause of the popularity of semantic segmentation as the preferred DL framework to be employed in advanced crack identification and inspection [37, 16, 38]. The main reasons for focusing on pixel-level segmentation are:

2.3.1 Correct Geometric Quantification

The greatest advantage of segmentation is that it yields a pixel-by-pixel map outlining the tiny detail and contour of cracks [22]. This kind of output makes it possible to extract and quantify crucial geometric parameters determining a crack's severity and impact potential, i.e.:

Length: A measure of the total length of the crack along its path, normally from the crack skeleton [23, 47]. Length is a key propagation indicator.

Width: The measurement of the width of the crack, often variable in its length and proportional to the likelihood of water and deleterious substance penetration [29, 47]. Segmentation allows for measurement at multiple points or averaging of the width.

Area and Density: Calculation of the total area of cracks in an area or crack density (crack area per unit surface area) provides a quantitative measure of surface distress [47].

Topology and Shape: Segmentation masks reveal the subtle shape, including branching patterns, intersections, and network structures (e.g., alligator cracking), which offer information beyond linear measurements [31, 10]. This morphological information is crucial in the interpretation of crack type and potential causes [16].

2.3.2 Better Localization and Mapping

Compared to bounding boxes which may contain a lot of background or patch classification which only defines areas impacted, segmentation defines exactly which pixels constitute the crack. High-accuracy spatial mapping is essential for:

Targeted Repairs: Guiding repair processes directly to where damage has happened, optimal material efficiency, and effectiveness.

Crack Propagation Monitoring: Segmentation masks of inspections carried out at different points in time may be compared to accurately monitor growth (length, width, or area increase) and propagation direction of cracks, providing useful information for remaining service life estimation [19].

2.3.3 Objective and Consistent Assessment

A transformative advantage of semantic segmentation is that it entails a fact-based, objective description of cracks and thus transcends the inherent subjectivity and inconsistency of human visual inspection [21, 16]. Human inspections are influenced by the well-established between- and within-inspector inconsistency due to human factors like experience and fatigue, which renders the data unreliable for the critical process of monitoring defect growth with time. In sharp contrast, a deep learning model trained is a deterministic model and therefore for any given input image, the output segmentation mask is reproducibly accurate and unbiased by humans. This consistency lays down a repeatable and auditable baseline for every inspection on which real quantitative longitudinal analysis can allow engineers to actually quantify crack growth, calculate deterioration rates, and make data-based decisions for risk-informed asset management.

2.3.4 Foundation for Advanced Analysis

The highly accurate crack maps from segmentation may be used as input to further analysis, such as Finite Element Modeling (FEM) in order to determine stress concentrations near the crack tip or correlation against other sensor readings in a bigger SHM setting.

2.4 Enabling DL Architectures for Crack Segmentation

The effective application of Deep Learning to the challenging problem of pixel-level crack segmentation depends on the development of custom neural network architectures that are capable of both understanding high-level semantic context (i.e., identifying crack-like patterns) and precisely localizing such features in spatial locations. Generic CNNs are actually excellent feature extractors, but they increasingly give up spatial resolution using pooling or strided convolutions, and their final fully connected layers give up all spatial information. To enable dense prediction (annotate all pixels), there needed to be special architectures. The dominant paradigm that emerged is the encoder-decoder network [39, 37, 40].

2.4.1 The Encoder-Decoder Paradigm

This structure forms the backbone of most modern segmentation models.

Encoder

This part typically copies a base classification CNN (e.g., VGG [56], ResNet [57], EfficientNet [40], etc.). Its purpose is to pass the input image through a sequence of convolutional and downsampling (pooling or strided convolution) layers. With the reduction in spatial resolution, typically more feature channels are utilized, where the network can successively identify higher-level and more abstract semantic features and increase the receptive field (the area in the input image influencing a particular feature). Hierarchical feature detection is important while identifying the context and crack-like features from cluttered backgrounds. Encoders learned from huge image datasets like ImageNet are utilized in most instances using transfer learning, capitalizing on strong general visual features learned on the prior dataset, especially useful where labeled data for cracks is limited [21, 42, 14].

Decoder

The decoder is responsible for mapping the encoder's low-resolution, semantically meaningful feature maps and progressively upsample them to the original image size. Decoder constructs the spatial information necessary for pixel-level classification. Simple upsampling methods (e.g., bilinear interpolation or simple transposed convolutions) confined to encoder's lowest features have a tendency to

output coarse segmentation masks with ill-defined edges, too rough to precisely define thin cracks [22]. It is therefore necessary that multi-level information combining has high-level decoder structures and procedures.

2.4.2 Notable Architectural Advances and Designs

Some notable architectural advances have been constructed and adapted to further boost the performance of encoder-decoder networks in segmentation tasks, some specifically aimed at addressing crack detection problems:

Fully Convolutional Networks (FCN)

The paper [53] had established that end-to-end dense prediction was possible by substituting the last fully connected classification CNN layers with 1x1 convolutional layers. Central to FCNs was the discovery of skip connections, where shallow encoder feature maps (more spatial detail preserved) are concatenated with upsampled feature maps from deeper layers in the decoder. This blending of coarse semantic information and high spatial information was a revolutionary step and permitted more precise localization of segmented objects.

U-Net

Highly impactful, particularly within medical imaging when boundary demarcation must be very accurate and adopted regularly to detect cracks [24, 54, 15, 40]. U-Net is a mirror encoder-decoder network with extremely widespread adoption of deep concatenative skip connections. Properties of an encoder layer are, instead of summation, concatenated channel-wise with the upsampling corresponding features in the decoder and subsequently convolved again, as shown in Figure 2.1. This gives the decoder path much richer sets of high-resolution features of the encoder, greatly improving its ability to reconstruct high details and outline objects with intricate boundaries, e.g., thin cracks [58].

Nested UNet (UNet++) architecture

Re-designed skip pathways This is a major development in the original U-Net architecture, where innovations were made to improve segmentation performance.

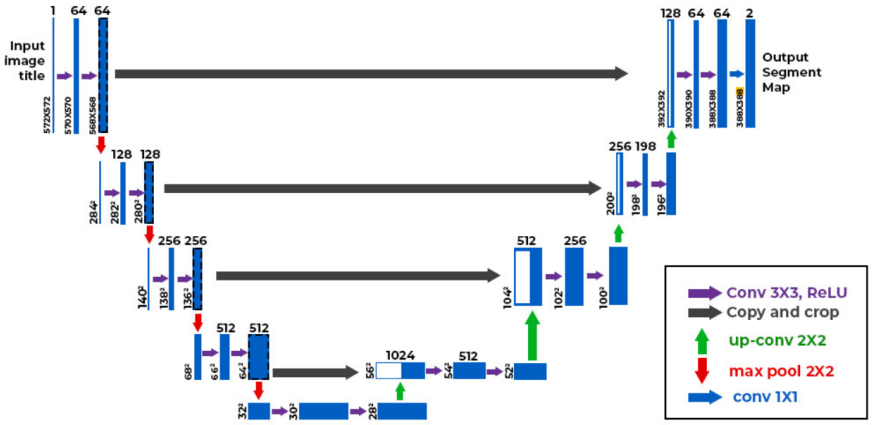

Figure 2.1: U-Net Architecture.

This development involved redesigning the skip pathways for the encoder-decoder subnetwork connection. In U-Net, this connection occurs directly from the encoder to the decoder, whereas in UNet++, these connections are subject to a dense convolution block, the number of which depends on the level of the hierarchy. The skip pathways between nodes in the dense blocks $X^{0,0}$ and $X^{1,3}$, for example, pass through three convolution layers, where the outputs of the previous layer of the same node are merged with the corresponding calibration outputs of the node with the lowest heuristic before each successive convolution layer. These skip pathways were strategically designed to bridge the semantic gap between the feature maps produced by the encoder and decoder. The basic principle is to make the optimization process easier when the merged feature maps are more semantically similar. The skip pathway is formulated as follows: Let $x^{i,j}$ denote the output of node $X^{i,j}$, where i indexes the down-sampling layer along the encoder, and j indexes the convolution layer of the block along the skip path [3]. The set of feature maps represented by $x^{i,j}$ is calculated as follows:

$$x^{i,j} = \begin{cases} \mathcal{H}(x^{i-1,j}), & j = 0 \\ \mathcal{H}\left(\left[x_k^{i,j-1}\right]_{k=0}^{j-1}, \mathcal{U}(x^{i+1,j-1})\right), & j > 0 \end{cases} \tag{2.4.1}$$

$\mathcal{H}(\cdots)$: The convolution operation function followed by the activation function.

$\mathcal{U}(\cdots)$: Denotes the sampling layer.

$[\cdots]$: Denotes the pooling layer.

- From the previous layer of the encoder, there is one input at level j = 0.
- From the encoder subnetwork, there are two inputs at two successive levels at level j = 1.

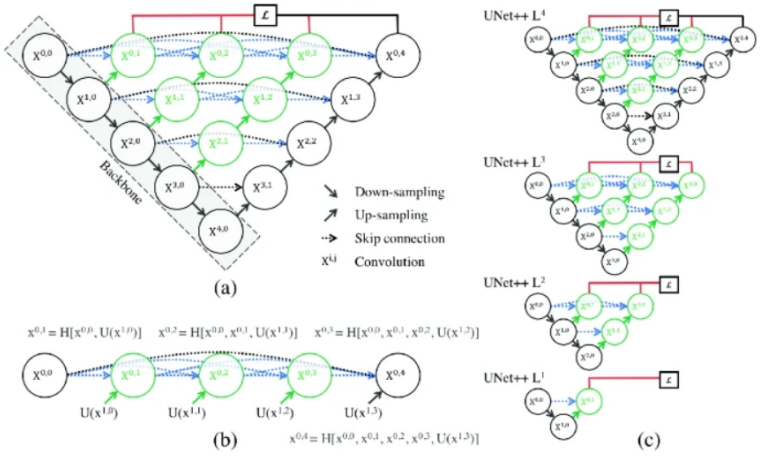

Figure 2.2: Nested UNet (UNet++) architecture [3].

- Nodes at level j > 1 receive two inputs j + 1, of which j inputs are the outputs of previous j nodes in the same skip pathway .
- The last input is the optimizer's output from the lower skip pathway.
- Using a dense convolution block along the overflow path causes all previous feature maps to accumulate and reach the current node. Figure 2.2 further illustrates how the feature maps travel through the overflow path of U Net++ [3].

Deep supervision U-Net++ with deep supervision enables the model to operate in two modes: Accurate mode: The mean is mathematically extracted for each output of the segmentation branches; Fast mode: A single segmentation branch enables the selection of the final segmentation map. This determines the extent to which the model is pruned and increases speed. Figure 2.2 shows how the branch is selected in Fast mode. Overlapping skip pathways makes the feature maps highly accurate at multiple semantic levels and amenable to deep supervision. A loss function is added to each of the four levels in $\{X^{0,j}$,j $\{1,2,3,4\}\}$, limited to a combination of the binary cross-entropy and the Dice coefficient, as shown in the following equation:

$$\mathcal{L}(Y, \hat{Y}) = -\frac{1}{N}\sum_{b=1}^{N}\left(\frac{1}{2}Y_b \log \hat{Y}_b + \frac{2Y_b\hat{Y}_b}{Y_b + \hat{Y}_b}\right) \tag{2.4.2}$$

Y_b: the flattened ground truths of the b^{th} image.
$\hat{Y}_b$: the flattened predicted probabilities.
N: Batch size.
In short, U-Net differs from UNet++ in 3 ways:

- The semantic gap between the encoder and decoder feature maps is bridged by convolutional layers on the skip paths (shown in green).
- Gradient flow is improved by providing dense skip connections on skip pathways (shown in blue).
- Pruning and improving the model by providing deep supervision (shown in red), or in the worst case, achieving performance similar to using only one loss layer [3].

SegNet

SegNet architecture focuses on memory and computation efficiency in the decoder [4, 47]. In max-pooling operations during the encoder, SegNet retains the indices (locations) of the max values. In the decoder, it utilizes the retained indices to perform non-linear upsampling and restore the features to their original locations before convolution, as shown in Figure 2.3. This does not learn the upsampling filters but may be less precise in restoring the fine details compared to the concatenation mechanism of U-Net.

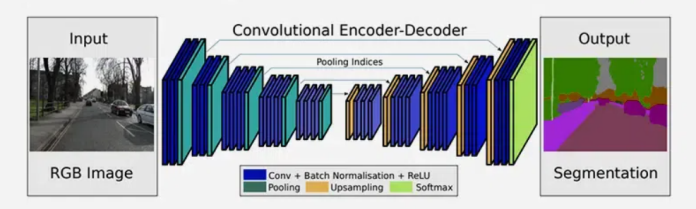

Figure 2.3: SegNET Architecture [4].

Atrous (Dilated) Convolution and DeepLab

In trying to get multi-scale context without losing spatial resolution too prematurely, DeepLab models [59]depend heavily on atrous convolution. Atrous convolution is all about incorporating gaps (controlled by a dilation rate) into standard convolution filters, effectively expanding the filter receptive field without increasing the number of parameters or reducing the output resolution, as shown in Figure 2.4. The Atrous Spatial Pyramid Pooling (ASPP) module carries out several parallel atrous convolutions with different dilation rates (and sometimes global average pooling) on the bottom encoder feature map. This allows the

network to analyze features of different scales simultaneously in an end-to-end manner, which is particularly helpful for the detection of cracks of different sizes [22].

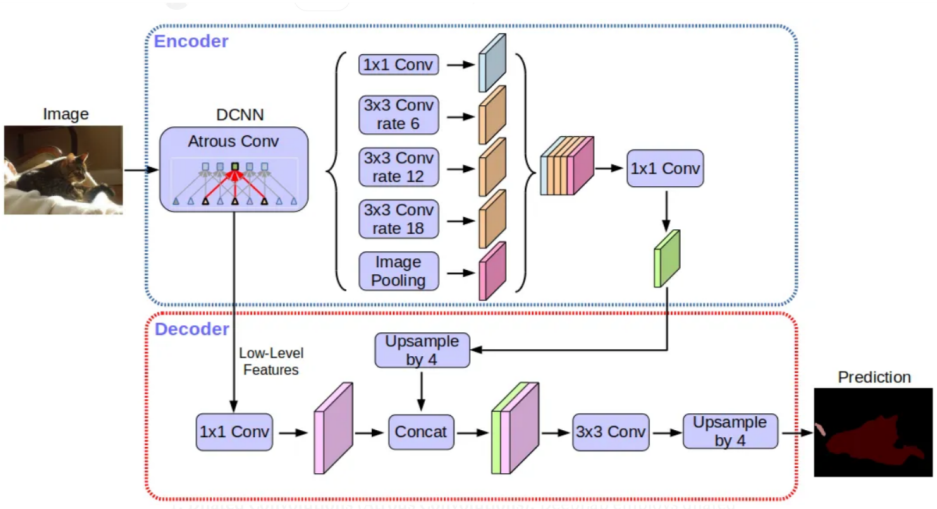

Figure 2.4: DeepLab Architecture.

Dense Connectivity (e.g., FC-DenseNet)

As a variant of DenseNets for classification [60], fully convolutional variants including FC-DenseNet [61] use dense blocks in which the feature maps from all previous layers are consumed by each layer while offering its feature maps to all subsequent layers within the block. This encourages reuse of features, improves gradient flow (less vanishing gradients for deep networks), and can lead to parameter-sparsity models. These are advantages that have been leveraged in some crack segmentation networks [37, 24], An illustration of the FC-DenseNet architecture is shown in Figure 2.5.

Attention Mechanisms and Transformers

Understanding that not all spatial locations or all features are equal, attention mechanisms allow models to choose to consider only the most relevant information.

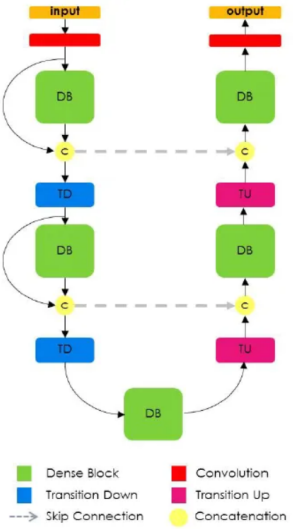

Figure 2.5: FC-DenseNet Architecture.

These can be added to CNNs as modules with regard to channel or spatial relations [15, 43]. More abstractly, Vision Transformers (ViTs) and their variants process images by dividing them into patches and using self-attention to capture patch relations [62]. These models like the Segment Anything Model (SAM) [63] or specific segmentation transformers (e.g., those based on SWIN transformer blocks [15]) utilize this capability to segment so that they can capture long-range relationships and global context, which could be useful to segment highly big or inter-connected crack networks. Hybrid architectures fusing CNN encoders (for local details) and transformer decoders or blocks are also becoming increasingly popular [39].

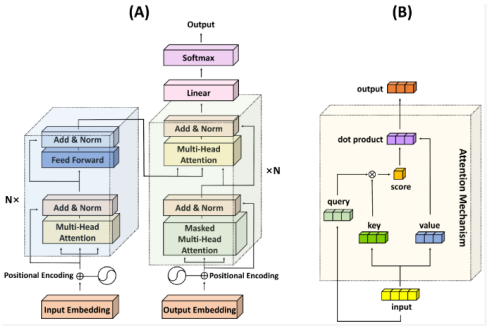

Figure 2.6: Illustration of the transformer architecture and the attention mechanism. (A) Transformer structure; (B) Attention mechanism [5].

Lightweight and Efficient Designs

Computational efficiency is a prerequisite for realistic deployment, particularly on mobile devices (vehicles, drones) or edge devices. This has led to the creation of light-weight models with an emphasis on speed and lower resource utilization, typically through techniques like depth-wise separable convolutions (employed in MobileNet [64] and adapted in [47, 22]), Ghost modules [46], group convolutions, or domain-specific blocks like those employed in SDDNet [22] and AnomalySeg [43].

2.4.3 Comparison of Deep Learning Segmentation Architectures for Crack Detection

The choice of a particular deep learning architecture is determinantal during crack segmentation framework design. Dominant models like FCN, U-Net, SegNet, DeepLab, and more recently transformer-based architectures are each design philosophies competing for optimal semantic understanding and accurate localization. Understanding their underlying principles and inherent trade-offs, as presented in Table 2.2, is critical to results interpretation and future improvement:

2.5 Related Works

Automated detection of cracks in civil infrastructures has garnered considerable interest in research and practice to improve the efficiency, objectivity, and safety of structural health monitoring. Traditional detection processes primarily involved human intervention or classical image processing techniques for cracks including threshold images, edge detection, and so on, which were limited in robustness, scalability, and adaptability to different real-world situations in the pavement (e.g., differences in light conditions, shadows, textures, etc.) [42, 28, 14, 65]. Deep learning has become an important means to address some of these challenges; in particular, Convolutional Neural Networks (CNNs) offer immensely useful computational frameworks for crack detection. The existing deep learning strategies for crack detection can be categorized on the basis of the main output and input structure: image classification, object detection, and semantic segmentation.

Table 2.2: Comparison of Architecture Families for Crack Segmentation

Architecture Family	**Core Distinguishing Feature / Principle**	**Key Strength (Relevant to Cracks)**	**Key Potential Weakness (Relevant to Cracks)**
FCN (Baseline)	End-to-end convolutional; Basic skip connections (sum).	Foundational; Enables pixel-level output.	Often produces coarse/blurry boundaries.
U-Net Family	Encoder-Decoder with Concatenative Skip Connections.	Excellent Fine Detail Preservation (good for thin cracks).	Standard receptive field might limit broad context understanding.
SegNet Family	Uses max-pooling indices from the encoder for non-linear upsampling in the decoder.	High memory and computational efficiency, especially in the decoder.	Can be less precise in restoring fine details and sharp boundaries compared to concatenation-based methods.
DeepLab Family	Atrous (Dilated) Convolutions & ASPP.	Strong Multi-Scale Context (good for varying widths).	Can sometimes miss the very finest details or have artifacts.
Transformer / Hybrid	Self-Attention Mechanisms.	Captures Long-Range Dependencies / Global Context.	Computationally heavier; potential loss of local detail if not hybrid.
Lightweight Designs	Replaces standard convolutions with computationally efficient operations (e.g., depth-wise separable, group convolutions) to drastically reduce model parameters and FLOPs.	Enables real-time inference on mobile or edge devices (drones, vehicles) with limited computational power and battery life.	May exhibit a slight reduction in segmentation accuracy, especially for very fine or complex low-contrast cracks, due to a more constrained feature extraction capacity compared to larger models.

2.6 image classification

Early deep learning applications to crack detection predominantly framed the problem as an image classification task. This methodology typically involves dividing larger images into smaller patches, which are then fed into a CNN to classify each patch as either containing a crack or being defect-free.

Zhang et al. [32] were one of the earliest efforts to successfully deploy deep convolutional neural networks (DCNNs) to classify patches of images containing pavement cracks, and their performance demonstrated that DCNNs can perform better than conventional machine learning approaches such as support vector machine (SVM) and boosting methods. Zhang et al. [32] had evidence of showing that DCNNs learn discriminative features from raw pixels of input images. Cha & Choi [28] developed the first DCNN-based system for pavement crack detection using wearable images. They trained their model, where they reported using 40,000 concrete image patches (256 x 256 pixels) of various cracks on pavement, and found their model achieved roughly 98% accuracy. Analyzed too, was the application of the FCN on larger images, as well as slipping window approach on the images revealed a good classification/detection results. They reported their DCNN systems outperformed traditional edge detectors, primarily Canny and Sobel. Furthermore, Cha and Choi concluded that they DCNN with software toolbox was still efficient under real conditions where there is variety in lighting and even when cracks were thin.

Given the enormous amounts of data that need to be collected to train deep CNNs from scratch, transfer learning was a popular option. Da Silva et al. [42] accomplished this by fine-tuning the VGG16 architecture used video of 3,500 concrete surfaces and reported 92.27% accuracy for its application in Unmanned Aerial Vehicle (UAV) inspections. Their research acknowledged the importance of some factors such as the learning rate, the number of nodes included in the last fully connected layer and the size of training dataset. Chen et al. [21] further demonstrated the effectiveness of transfer learning when compared to a CNN trained from scratch that achieved 89% accuracy with their detection of building surface cracking model after training on a dataset of 3,600 images compared to a transfer approach using ResNet101 which produced an accuracy of 94%. Rajadurai Kang [14], explicitly used AlexNet for crack detection, and constructed an image dataset that included "no-cracks" and "crack" scenarios. They used transfer learning to fine-tune the weights of the architecture and changed the classification layer to produce two output classes. The fine-tuned AlexNet was intended for image classification, they trained the architecture using optimization algorithm stochastic gradient descent with momentum and achieved 99.9% accuracy, precision, recall and F1 scores on the validation and test datasets. AlexNet demonstrated robustness

in performance when tested against a very different dataset although there was a small drop off in performance in a couple of areas due to user obstruction example shadow and surface roughness.

While effective for identifying the presence of cracks within regions, these patch-based classification methods generally require a subsequent scanning mechanism for full surface inspection and do not inherently provide precise localization or detailed morphological information about the cracks, which is crucial for quantitative assessment.

2.7 Object Detection

Object detection frameworks that predict bounding boxes around defects have been used more regularly for crack detection in a more precise way than pure patch classification (which involves classifying defects in an image as one or more labels - no spatial information is given). Object detection methods provide a label and also spatial coordinates of defects.

The You Only Look Once (YOLO) family of models has been widely adopted due to the trade-off between speed and accuracy of real-time solutions. Yu [52] proposed YOLO V5s-based solution for detecting concrete cracks using 3500 manually labeled images, which served to improve the original YOLO V5s model by using Otsu thresholding to remove background noise and K-Means to predict the best initial anchor box sizes. After applying these techniques, Yu's [52] article reported an average precision (AP) of 84.37%, average recall (AR) of 76.01% and F1-score of 79.97%.

Dong et al. [46] advanced YOLO based detection significantly with a variation of the original lightweight YOLOv8-based algorithm called YOLOv8-CD. They developed an improved model architecture based on YOLOv8, combining a Large Separable Kernel Attention (LSKA) module, integrating visual attention networks and large convolutional attention and effectively capturing crack and local feature information to adapt for fracture susceptibility and slender shapes. The authors included the Ghost module into the YOLOv8 backbone for an efficient extraction of features, and they replaced the original convolution structure in the neck of the network with GSConv and a VoV-GSCSP module, reducing floating point operations while still achieving accuracies. YOLOv8-CD was tested (on the RDD2022 dataset and Wall Crack dataset) and ultimately achieved substantial mAP50 increases of 15.2% and 12.3%, respectively, over a baseline YOLOv8n model. They reported a detection speed of 88 FPS. The work by Fu et al. [65] contextualizes the relevance of their work by reviewing the development progression of object detection from R-CNN and its derivations (Fast R-CNN, Faster R-CNN), and other single-stage detectors including YOLO and SSD, and highlighted the constant pursuit of

improved speed and accuracy.

Even though object detection provides us with advantageous localizations with bounding boxes, such rectangular outputs do not always represent the intricacies, linearity, complexity, and usually poor geometric representation of a crack as accurately as pixel-based segmentation methods.

2.8 Semantic Segmentation

For applications demanding detailed morphological information and precise delineation of crack boundaries, semantic segmentation techniques, which classify each pixel in an image, have become the standard. These methods generate a pixel-wise mask identifying crack regions, enabling more accurate quantitative analysis.

Originally designed for biomedical image segmentation, the U-Net architecture [54] has been adapted for crack detection reliably and successfully due to the inherently strong encoder-decoder structure and skip layers that combine multi-scale features and maintain spatial information. Ronneberger et al. [54] demonstrated U-Net's abilities for image segmentation with very little training data in combination with a strong level of data augmentation, such as elastic deformations. Cheng et al. [66] used U-Net for pixel-wise crack detection, being distinct to process the entire image rather than using patches and also introduced a new cost function based on the distance transform to give pixel level weights for handling class imbalance. Their U-Net performed very well, with over 92% pixel-wise segmentation accuracy for the two public road crack datasets tested (CFD and AigleRN). Similarly, Gao [67] constructed a (U-Net) approach for road and tunnel crack detection using data taken via a mobile mapping system. Following which he focused on improving accuracy through new parameter settings, fine-tuning, class imbalance treatment, and post-processing for vectorization to represent crack length and width.

U-Net and other Fully Convolutional Network (FCN) based architectures are the subject of recent interest mostly focused on variations or improvements. Liu et al. [6] proposed "DeepCrack", a deep hierarchical (without fully connected layers) FCN architecture based on FCNs and Deeply-Mocked Networks (DSN). The motivation for DeepCrack is to learn and combine multi-scale and multi-level features from each stage of the convulutional path (low to high), while also applying a DSN approach by having integrated direct supervision at each of the convolutional stages. The authors employed guided filtering and Conditional Random Field (CRF) techniques as post processing steps to adjust the predictions they made using their new benchmark (537 images) dataset, achieved a mean Intersection over Union (IoU) of 85.9% and best F-score of 86.5%.

Zheng et al. [47] established a lightweight technique for detecting cracks in

bridges, extending their model from SegNet (another variant of encoder-decoder architecture) with bottleneck depth-separable convolutions and residuals. This development has been created to enhance efficiency and robustness. Their model achieved increases in accuracy to 97.95% and MIoU to 77.76% scores when compared to other methods such as DeepCrack and CrackU-net on their dataset. Li et al. [24] were focused on pavement cracks and provided a detection algorithm based on a densely connected and deeply supervised network. They used densely connected layers to support feature propagation and feature reuse, and deeply supervised modules to extract more prominent features at multi-scale levels. Their major contribution was a class-balanced cross-entropy loss function. They demonstrated a superior performance to other approaches specifically dealing with the imbalance between the small area of crack pixels and the larger area of background. Li et al. [24] assessed their model on three public datasets: AEL, Crack500, and Cracktree200.

Wang & Su [40] proposed a semi-supervised semantic segmentation network for detecting surface cracks and used a modified version of EfficientUNet as the backbone of their model. They used a student and teacher model (the teacher model weights updated by the exponential moving average of the student model.), and trained the network with both annotated and unannotated data while injecting noise for robustness. The authors focused on the amount of annotated data used; after using only a modest 60% of the annotated data, they achieved an F1 score of 0.6540 on their concrete crack dataset and 0.8321 using the Crack500 dataset, thus greatly reducing the amount of labeled data needed.

To address the issue of small or unusual flaws being detected, Song et al. [43] proposed "AnomalySeg," which was a refined U-Net. Instead of a normal residual module, their model introduced a hybrid residual module where the improved spatial attention mechanism (which featured multi-scale dilated convolutions) and a feed-forward neural network were combined to replace almost all of the down sampling layers of the encoder in order to preserve information regarding small imperfections. Dilated convolutions were also adopted in the decoder and a more generic hybrid loss function (Dice and focal loss) was proposed to overcome the small defect segmentation problem. AnomalySeg achieved better Dice coefficients on KolektorSDD, KolektorSDD2 and RSDD datasets using fewer parameters than any other generic segmentation benchmarks.

Tabernik et al. [68] presented a segmentation-based deep learning architecture to learn from a very small number of defective training samples (around 25 to 30) using their newly developed Kolektor Surface-Defect Dataset (KolektorSDD). Their two-stage architecture had a segmentation network (11 convolutional layers) for pixel-wise localization, and decision network on top to predict per-image defect presence, which surpasses comparator methods and commercial software in their

specific industrial product defect area.

Fan & Zou [69] proposed a new technology for road crack detection based on "deep dictionary learning and encoding networks (DDLCN)" along with a new activation function MeLU. The DDLCN replaces traditional convolution layers with dictionary learning encoding layers - they use this DDLCN within an improved Mask R-CNN framework (an instance segmentation model combining object detection and pixel level mask). This proposed technology was not just about detection but about also analyzing specific characteristics of cracks, measurements and features, etc.

Fu et al. [65] improved the DeepLabv3+ architecture specifically for bridge crack semantic segmentation. They altered the structure of the DeepLabv-3+ network by adding a "densely connected atrous spatial pyramid pooling (ASPP)" module. The modification allowed the network to make denser pixel sampling and to extract detail features better, by concatenating feature maps from atrous convolution with small rates to be input to lots with large atrous rates, thus maintaining a larger receptive field with the total number of parameters of the model identical. The improved model achieved an average intersection ratio of 82.37% on the bridge crack dataset they collected.

Li et al. [70], Shenyang Jian Zhu University proposed a lightweight model called "Mini-Unet" specifically designed for the detection of tunnel lining cracks. This lightweight model builds off of the U-Net architecture by utilizing fewer downsampling steps, and replacing some of the normal convolution layers with depthwise separable convolution (DSConv) layers to increase efficiency. A hybrid loss function that included Dice loss and cross-entropy was also employed to address the imbalance between the backgrounds and cracks categories. Mini-Unet performed well, yielding a mean IoU of 60.76%, mean precision of 84.18%, and a mean inference speed (FPS) of 5.635. The Mini-Unet outperformed a number of mainstream models in terms of efficiency.

Goo et al. [39], introduced "Hybrid-Segmentor," an encoder-decoder model consisting of a CNN model (ResNet-50) path for fine-grained local features and a Transformer model (SegFormer concepts: Overlapping Patch project, Efficient Self-Attention, and Mix-FFN) path for global context to create a hybrid encoder-decoder architecture. The entire feature maps from both paths were straight-forwardly fused at multiple levels before the simple decoder. The hybrid model was trained using a BCE-DICE loss function and achieved state-of-the-art results (0.971 accuracy, 0.770 F1-score, and 0.630 IoU) on a large refined dataset by combining and augmenting 13 open-source crack datasets.

These varied types of semantic segmentation approaches clearly illustrate the trend of more accurate, efficient, and robust pixel-level crack detection applications that deal with all sorts of complex crack morphology and imaging conditions, while

often seeking to minimize dependancy on massive amounts of labelled data.

The previous literature reviewed a clear trajectory in deep learning for crack detection. These developments move increasingly away from patch-image classification and bounding-box object detection to semantic segmentation at the pixel level. Although patch-image classification validated the use of deep learning, and bounding-box object detection enhanced the localization of cracks, pixel-level semantic segmentation is the most accurate representation of crack morphology, particularly in using architectures like U-Net [54] and SegNet-based models [47] and the various modified versions of U-Net [66, 67, 40, 43].

Conclusion

In this chapter, we established semantic segmentation as the optimal approach for crack detection, mainly due to its unique ability to provide the pixel-wise precision needed for quantitative geometric analysis. The review of enabling architectures, found the most successful approach to be the encoder-decoder model, and more importantly those architecture instances such as U-Net and UNet++ with proposed advanced skip pathways, as the most successful paradigm to balance the crucial relationship between deep semantic feature extraction and spatial localization at a fine-grained spatial extent. This methodological choice is supported by a noted in the literature with a tendency to dwell towards a pixel-wise analysis. We have now presented a robust methodological and architectural rationale, and we will move to demonstrate the empirical application and evaluation of the proposed segmentation model.

Chapter 3

Our Solution

Introduction

This chapter provides the details of the practical application of the deep learning-based crack segmentation model. First, this chapter discusses the dataset from which the data was obtained, data preprocessing, and data-augmentation methods used to improve model generalization. The chapter also talks about the proposed deep learning architecture and the training pipeline. Next, describes the experimental environment (the programming tools and computational environment used to support the development and evaluation of the model). Next, performance evaluation metrics, along with comparative results against other state-of-the-art approaches. Finally, a simple but efficient crack detection system has been created and demonstrated to showcase the real-world applicability of the model. This experimental framework will help to conclude the theoretical exploration and theoretical development of the proposed approaches, while also addressing how to transition to software deployment within structural health monitoring.

3.1 Dataset

For this study, we utilized the DeepCrack dataset [6], a well-respected open-standard benchmark for the detection of cracks via image segmentation. There are a handful of other publicly available datasets for this purpose, such as CrackTree200 [10], but we selected DeepCrack because it contains high-quality pixel-level annotations, which are instrumental in training and testing good semantic segmentation models.

The data set consists of 537 image-mask pairs. Input images are RGB photos in

Figure 3.1: The first row is the original image which is the DeepCrack data set, and the second row is the artificial mark of the crack.

the JPEG format. The compression algorithm for this format is lossy compression, which is very effective at storing exact visual data but perhaps introduces subtle artifacts. Every image has an accompanying binary crack mask, and it is employed as training truth ground. Importantly, these masks are supplied in the PNG format. The use of PNG is intentional and important, for it is a lossless compression format. This means that the exact, pixel-wise position of all cracks is maintained with perfect fidelity, free from any compression artifacts that would potentially debase the quality of the ground truth data. The dataset is split into two subsets, with a training set of 300 images and a test set of 237 images, and each image is at a fixed resolution (544 × 384 pixels). Each image has an associated binary mask that traces the spatial distribution of crack pixels, as shown in Figure 3.1. Table 3.1 lists that the predominant pixel represents the background (no crack) so there is a clear class imbalance. Images consist of predominantly asphalt and concrete surfaces; the crack width varies between 1 pixel and 180 pixels, which makes multi-scale crack segmentation a difficult task [6].

Table 3.1: The percentages of crack pixels and non-crack ones [6].

	Crack pixels (%)	**Non-crack pixels (%)**
Training	2.91	97.09
Test	4.33	95.67
Total	3.54	96.46

3.2 Environment

3.2.1 Working Environment

The model's development, training, and subsequent evaluation were conducted utilizing the Colab Pro environment, which provided critical computational advantages. Access to high-performance Graphics Processing Units (GPUs), such as the Nvidia T4 or P100, was instrumental in accelerating the intensive computations inherent in deep learning model training. Furthermore, the provision of substantial Random Access Memory (RAM), typically 25GB or greater, and extended runtime limits ensured the uninterrupted and efficient execution of prolonged, resource-intensive processes ran seamlessly and efficiently during the course of this work.

3.2.2 Programming Language

Python: Python is the most popular high-level programming language recently. It allows developers to write less code compared to other languages. Its strength lies in its support for diverse programming paradigms and its rich, extensible library [71]. Among its most important features and advantages are:

- Easy to use and learn: Its syntax is simple.
- Free and open-source: Anyone can use it.
- Multi-platform and portability: It runs on various operating systems.
- Supports both procedural and object-oriented programming: It is flexible in building applications.
- Interpreted language: Python instructions are not converted to machine code for execution.
- Extensibility and embedding: Python can be integrated with other languages or embedded into other applications.
- A comprehensive standard library: It provides ready-made tools and templates for a wide range of tasks.

3.2.3 Libraries

- **PyTorch:** Python uses the PyTorch library, which is based on the original Torch framework, developed by Facebook. It features dynamic graphs, simplicity, and is more advanced than TensorFlow, allowing developers to easily modify and debug their software. The gradient ranges from the data tensor to the network abstraction levels, from tensor to variable to nn.Module, respectively [72].

- **TensorFlow:** TensorFlow is used in various scientific fields. It is a framework for defining and running computations with partially defined computational objects that ultimately yield a value, called tensors. TensorFlow visualizes computational graphs better, reduces errors by 50 to 60 percent in machine learning, and features parallel computing for executing complex models. It manages a series of libraries supported by Google, and offers faster updates and the latest features through frequent releases. TensorFlow is useful for speech and image recognition, text-based applications, and video detection using time series analysis. It is high-performance, as attested to by 35,000 comments from a community of 1,500 contributors [73].

- **NumPy:** NumPy is the core numerical computation library in Python. Its strength lies in its high-performance N-dimensional array and a comprehensive set of functions and tools for efficient operation. NumPy overcomes the slowness of Python numerical operations by enabling matrix-oriented computation and pre-compiled vector operations. Its need for large-scale data analysis is essential for many applications. It is the basis for other scientific and analytical libraries such as SciPy and scikit-learn. It may be a suitable alternative to MATLAB after integration with other libraries such as Matplotlib and SciPy. It is widely popular and supported by many contributors and users [73].

- **Matplotlib:** Matplotlib is a powerful Python library designed to create effective and efficient visualizations and graphs. It has a broad user base and contributors, making it an ideal choice for data visualization. Its graphs can be easily integrated into various applications thanks to its object-oriented interface. It is the best free and open-source alternative to MATLAB for graphing, supporting various operating systems and output formats. Interaction with Matplotlib can be simplified by using the Pandas library as an interface, as it is highly memory-efficient and performs well during execution. Matplotlib is used for studying correlations between variables and representing confidence, identifying outliers through scatterplots, and analyzing data distributions for quick and in-depth understanding [73].

- **Scikit-learn:** Scikit-learn is a popular Python machine learning library that seamlessly integrates with other libraries such as NumPy and Pandas. It is considered the most prominent and preferred library for implementing classic machine learning algorithms, as it is built on the SciPy and NumPy libraries. It provides comprehensive support for most supervised and unsupervised learning algorithms. It can be used effectively for data exploration and analysis tasks, making it a valuable tool and ideal for beginners in machine learning [73].

- **OpenCV:** OpenCV is a popular open-source computer vision library, specifically designed for applications requiring real-time processing. Its architecture includes hundreds of dedicated computer vision algorithms. Its components include diverse modules such as image processing, video analysis, object detection, camera calibration, 3D modeling, and 2D feature frameworks [73].

- **Argparse:** argparse is used in most custom Python scripts, it is a module that provides easy-to-use command-line interfaces [74].

- **OS:** The OS in Python serves as an interface that allows programs to interact with the underlying operating system. Interacting with OS functions is made easier using the OS, enhancing software portability across different environments. The OS module features an OSError exception that it raises when errors occur during interaction with the OS. It is essential for many programming tasks and automated testing, and is used to locate specific files such as configuration files, test reports, and even test data files in formats such as Yaml and Excel. Although there are methods available that enable interaction with the OS other than the OS to call specific system functions, this negatively impacts code portability [75].

- **SyS Modele:** System-specific parameters and functions: This module allows access to variables used or maintained by the Python interpreter. There are functions with which sys interacts closely. This module is always accessible. Variables in this module are read-only unless explicitly declared [76].

- **Tqdm:** Tqdm is a Python library that provides fast and scalable progress bars for iterations and loops as well. tqdm makes it easy to track the progress of time-consuming tasks and is able to update the display bar by counting repetitions, calculating elapsed and remaining time. The library's name means "progress" in Arabic (taqadum,), and is an abbreviation for "I love you so much" in Spanish (te quiero demasiado)." It efficiently visualizes the overall progress to improve performance and provide clear, consistent visual information [77].

- **Glob:** In Python, global is shortened to glob, and is a useful part of the standard library, for searching for file pathnames that match a certain pattern. It includes two functions glob.glob() and glob.iglob() and is also useful for searching for text within files and csv files. It is simple, and glob library styles can be considered similar to regular expressions. It is easy to use to read similar files before analyzing them [78].

- **Shutil:** Backing up files and folders can be simplified using the shutil module. The code for using the function shutil.copy(source, destination) copies the file located at "source" to "destination." The code for shutil.copytree() implements the same logic as file copying but handles an entire folder instead of a single file. It's worth noting that shutil.copytree() doesn't retain all file metadata, such as owner and group, which can be important in system administration environments. To ensure proper execution, it's essential to schedule shutil function calls within Python scripts using the operating system's task scheduler [79].

3.3 Data augmentation

Effective training of deep learning models, especially for complex tasks like image segmentation, heavily relies on the quality, quantity, and diversity of the training data. Data augmentation techniques are employed to address these needs by synthetically expanding the dataset and exposing the model to a wider range of variations. Our process incorporates two key stages: firstly, the strategic generation and filtering of patches from original large images and their segmentation masks to form a refined base dataset. Secondly, this base dataset of image-mask pairs is then augmented through a series of controlled geometric and photometric transformations, carefully designed to increase sample count and enhance feature learning while preserving the essential spatial relationship between images and their masks.

3.3.1 Patch Generation and Filtering for Deep Learning Image Segmentation Datasets

We comparatively segmented large input images and their corresponding segmentation masks of size 544x384 pixels into comparatively smaller, fixed-size subregions (patches) 384x384 pixels using a sliding window method defined by PATCH_SIZE = 384 and STRIDE = 128. Overlapping patches (if STRIDE < PATCH_SIZE) can help maintain contextual information at patch boundaries and the enrichment of the dataset. This configuration yields two patches along the

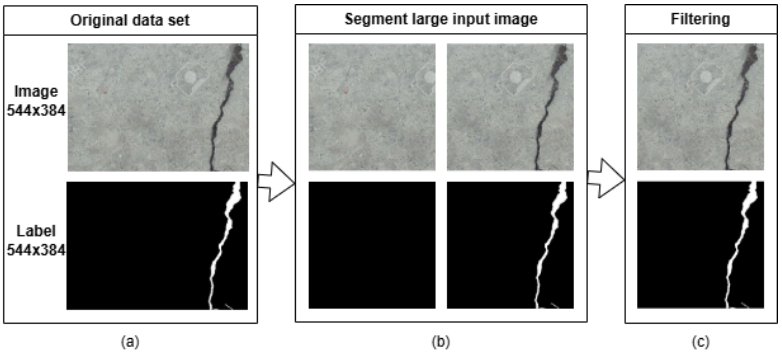

Figure 3.2: Data preprocessing pipeline for crack segmentation. (a) Original 544x384 image and label. (b) Generation of smaller, overlapping 384x384 patches using a sliding window. (c) Filtering to remove patches with no positive crack instances.

height dimension (floor((544 - 384) / 128) + 1 = 2) and one patch along the width dimension (floor((384 - 384) / 128) + 1 = 1), producing a total of two overlapping patches per original image. This is then followed by patches being removed where the mask has no positive cases (i.e., no cracks) and enriches the dataset with the proper samples to train the model.

3.3.2 Data Augmentation of Image-Mask Pairs

We expanded the image-mask pair dataset using TensorFlow image processing. This processing generates multiple augmented samples from each input sample in a deterministic manner by applying a series of random transformations, including geometric transformations (random vertical/horizontal flip, 90° rotation) and visual transformations (random brightness and contrast shift, Gaussian noise) to the images only. Importantly, the same set of geometric transformations is applied to both the image and its mask to maintain spatial congruence, which is essential for tasks like semantic segmentation. This effectively increases the size of the training dataset to 10,530 images and masks, improving model generalization and overfitting resistance by exposing the model to a larger set of visual inputs.

3.3.3 Data split

We systematically split the dataset containing images and their corresponding segmentation masks into subsets for training, validation, and testing, a critical step

for developing and evaluating a robust machine learning model. The dataset is randomly stratified into validated image-mask pairs based on specific ratios of 70% for training, 15% for validation, and 15% for testing. Finally, the data is prepared for model training and evaluation.

3.4 Proposed Model

U-Net Architecture: The U-Net architecture, originally designed for biomedical image segmentation [54], was adapted for crack detection in the present study. The U-Net architecture has a distinctive symmetric encoder-decoder structure, as shown in Figure 3.3, with the encoder path downsampling (in this case, from 256x256 pixels) the original image through a series of DoubleConv blocks followed by sequential 2x2 max-pooling layers, with each DoubleConv block containing two 3x3 convolution layers followed by Batch Normalization and a ReLU activation function. The downsampling path enables the extraction of progressively more abstract contextual features. The decoder path contains corresponding layers to the encoder path allowing the output spatial resolution to be the same as the original image. Similar to the encoder, the decoder, employs upsampling layers (bilinear interpolation, in this case) to map-up the Upsampled layers, which are then concatenated with their corresponding high-resolution feature maps from the encoder path via skip connections. Skip Connections were included to maintain and incorporate fine-grained details that were lost through the downsampling process. The final step in the architecture is a 1x1 convolutional layer (OutConv) that outputs the learned features into a binary crack segmentation map. U-Net is well-known for its performance in tasks that require accurate localization.

U-Net++ Architecture: U-Net++, an evolution of the U-Net architecture, was also employed and configured to process input images of 384x384 pixels. It aims to enhance segmentation accuracy by redesigning the skip pathways to be nested and dense. As does U-Net, the model follows a symmetrical encoder-decoder model in hierarchical feature extraction and accurate spatial reconstruction. Instead of straightforward, direct skip connections between the decoder and the encoder at the same hierarchical level, UNet++ introduces intermediate blocks of convolution on these paths. These blocks increasingly include feature maps from the encoder, added feature maps from increasingly deeper layers of the decoder, and outputs from previous hierarchical convolutional blocks at the same hierarchical level through pooling and additional convolution, as described in Figure 3.4. The close coupling is useful in segmentation as it reduces the semantic gap between low-resolution, semantically rich features of the decoder and low-level,

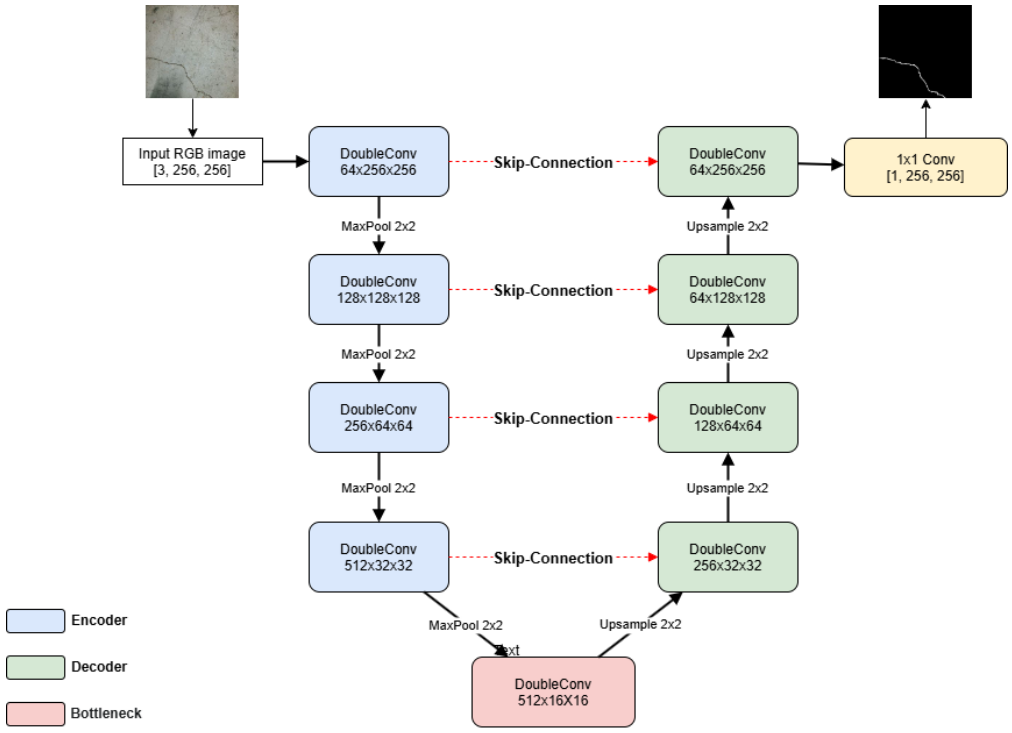

Figure 3.3: U-Net architecture for semantic segmentation-based crack detection.

high-resolution features of the encoder to result in improved object localization and boundary definition. Some other specific model enhancements suggested are residual ConvBlocks with skip connection within (self.conv(x) + self.skip(x)) that enables training deeper networks and better gradient flow for representing richer feature representations. Moreover, employing GroupNorm rather than batch normalization is also uniform in the performance across various batch sizes. This is particularly beneficial for high-resolution image segmentation where memory must most frequently run smaller batches. Generally, these enhancements assist in minimizing the construction of a superior model that is better equipped to recognize finer details and define segmentation masks, compared to the standard U-Net.

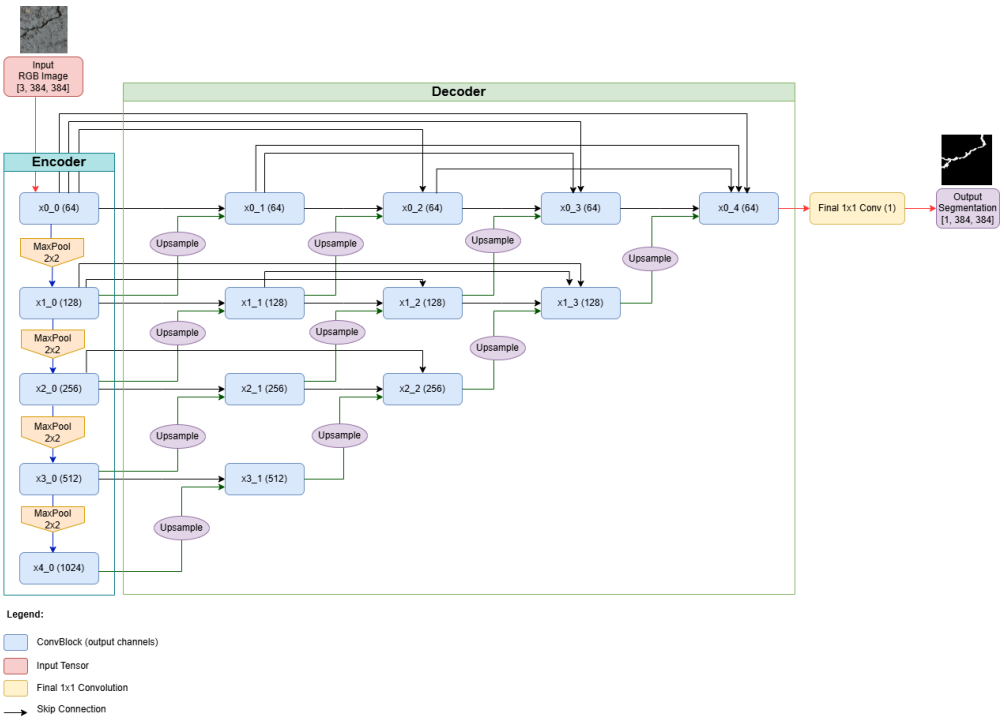

Figure 3.4: U-Net++ architecture for semantic segmentation-based crack detection.

3.5 Training the proposed model

The proposed model was trained for semantic crack segmentation via an iterative training routine for 70 iterations. The network was supplied with image batches and ground-truth masks at every training iteration. The model yielded logits that were mapped onto target masks by a binary cross-entropy loss function with

logits. This specific loss function is used as a best practice for numerical stability, as it combines the probability conversion (via a sigmoid function) and the loss calculation into a single, robust step. This information loss that measured the gap between prediction and reality was backpropagated through the network to update its internal weights by the Adam optimization algorithm. The training speed and memory consumption were optimized by automatic mixed precision. Model performance was also verified from time to time on a hold-out validation set with metrics such as Dice coefficient and IoU, and the best-validation-score model over epochs was retained as the final trained model.

Adam optimization algorithm: In the training of the semantic crack segmentation model, many of the training epochs relied on the Adam optimization algorithm to optimize the network's internal weights. Once the binary cross-entropy loss function computed the difference between the model's probabilities (or logits) and the ground-truth masks, this "information loss" began backpropagation to compute the gradients for each weight. Then, Adam (Adaptive Moment Estimation) would use these gradients for determining how to alter the weights. Adam adjusts the adaptive learning rate for each weight, which speeds up and slows down learning for different parameters, also uses "momentum" (an estimate for the first moment of the gradients) to accelerate progress in directions of consistent gradients, employs "RMSprop-like" scaling (an estimate for the second moment) to cause changes based on the variance of the gradients, and would eventually guide this network towards a point where we could adjust for semantic crack segmentation.

Automatic Mixed Precision (AMP) speeds up training and consumes less memory by performing many calculations for your model using faster and less precision, half precision (FP16). This makes particular sense on GPUs, where an AMP increase can show a substantial decrease in training times. PyTorch's autocast feature automatically identifies which operations can safely run in FP16, halving memory for activations and leveraging hardware acceleration. To maintain numerical stability and prevent issues like vanishing gradients common with FP16, *GradScaler* dynamically scales the loss up before backpropagation and then unscales gradients before optimizer steps, ensuring accurate weight updates while still benefiting from FP16's efficiencies.

3.6 Evaluation metrics

When training a deep learning model based on semantic segmentation, assessing its quality through evaluation metrics is crucial, and multiple metrics exist to achieve this. In our proposed crack detection model, we used Dice score, IoU score, F1 score, recall, precision and accuracy as evaluation metrics.

Figure 3.5: Training and Validation Loss curves over 70 epochs.

Avrage loss: Is a metric that quantifies how well the model's predictions match the ground truth labels on average, according to the specified *loss_fn* (e.g., Binary Cross-Entropy with Logits Loss). It is typically calculated on a validation or test dataset to evaluate the model's generalization performance.

$$\text{Loss}_{\text{pixel}} = -\left[G \cdot \log(\sigma(L)) + (1 - G) \cdot \log(1 - \sigma(L))\right] \tag{3.6.1}$$

Figure 3.5 visualizes the training and validation loss curves over 70 epochs. Both losses (BCEWithLogits) are consistently decreasing, with the validation loss of 5 closely tracking the training loss of 2, indicating good generalization of the model and the absence of significant overfitting.

Accuracy: It is the percentage of pixels that are correctly classified. This effectively evaluates the model [80]. In general, the accuracy statement can be modeled as:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.6.2}$$

Figure 3.6 visualizes the performance of a deep learning model for segmentation tasks over 70 training epochs. The model effectively improves its ability to correctly classify pixels over time, eventually reaching a very high level of accuracy on the validation data. While a high pixel accuracy 99.7% is generally positive, for segmentation tasks with potentially imbalanced classes (e.g., crack pixels being a small minority compared to background pixels).

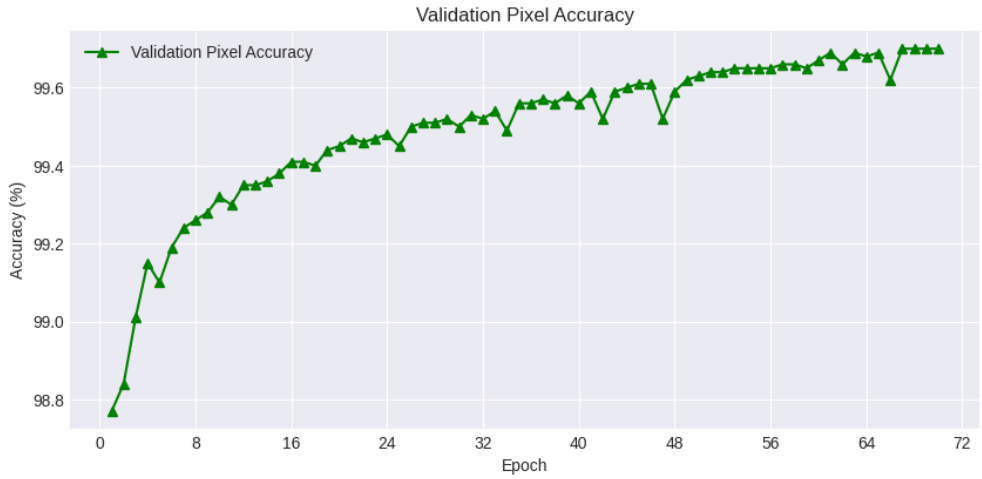

Figure 3.6: Pixel accuracy advances in model training stages.

Precision: It is the ratio of true positive pixels to all predicted positive pixels [80] and is expressed as follows:

$$Precision = \frac{TP}{TP + FP} \tag{3.6.3}$$

Recall: It measures how well the model can correctly identify and label all positive pixels:

$$Recall = \frac{TP}{TP + FN} \tag{3.6.4}$$

where:
TP: True Positive (correctly segmented positives pixels).
TN: True Negative (correctly segmented negatives pixels).
FP: False positive (incorrectly segmented positive pixels).
FN: False negative (incorrectly segmented negatives pixels).
High recall means that the model is able to predict the majority of relevant structures, without missing details.

Dice score: It measures how well the predicted segmentation matches the ground truth segmentation [80], and is formulated as follows:

$$Dice\ score = \frac{Area\ of\ Overlap}{Total\ Area} \tag{3.6.5}$$

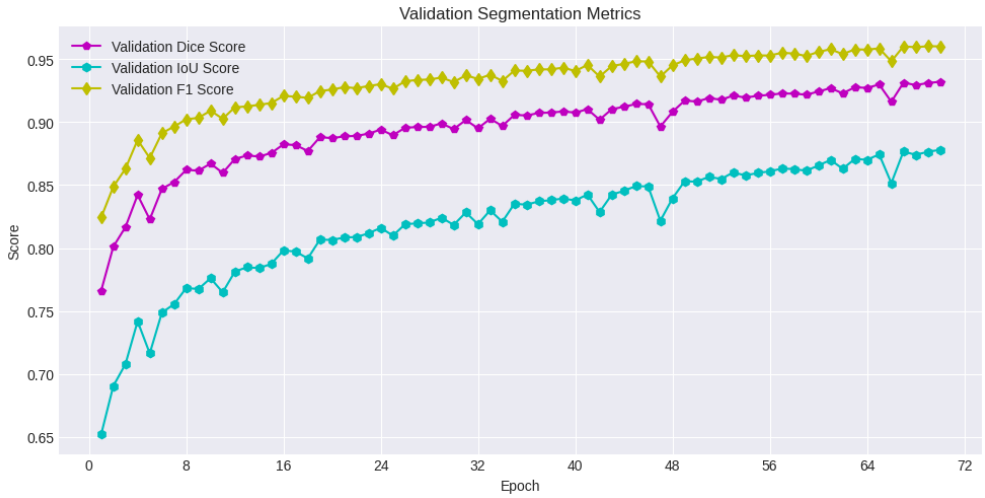

Figure 3.7: Performance Trends of Validation Metrics During Model Training.

It is most sensitive to small objects, and its agreement with the ground truth and the predicted result makes it useful for segmenting medical images and cracks.

IoU score: It represents the intersection ratio over union between the prediction and the ground truth [80], its expression is:

$$IoU\ score = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{3.6.6}$$

F1-score: It is a balanced measure of performance in imbalance scenarios, equal to the harmonic mean of precision and recall, and is formulated as follows:

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3.6.7}$$

F1 score explains the balance between precision and recall, this ensures that the model performs well in detection and segmentation without excessive false positives or negatives [80].

Firure 3.7 visualizes the performance of a deep learning model for segmentation tasks over 70 training epochs. It tracks three important validation metrics: the Dice score 0.9321, IoU score 0.8780 and F1 score 0.9602. Despite some fluctuations , all metrics improve overall, indicating that the proposed model is learning effectively and generalizing well on the validation set.

3.7 Results

The evaluation results show that the proposed UNet++ model achieves excellent performance on the crack segmentation task over 1579 test samples. The Dice score of 0.9338, and the IoU score of 0.8805 are strong scores, and indicate high spatial overlap or agreement between the predicted crack masks, and the ground truth masks, which is necessary for accurate localization in segmentation. The Pixel Accuracy is exceptionally high at 99.71%, which can probably be somewhat misleading, especially in imbalanced datasets (where non-crack pixels will dominate), however, the high Dice and IoU scores highlight that the model was effective in identifying the crack regions, and not simply classifying the background accurately. Moreover, the model also had a well-balanced Precision (0.9603), and Recall (0.9614), with a high F1 Score (0.9609) for the crack class, which means the model very rarely misclassifies background as crack (hence low false positives), and finds the majority of cracks actually present (hence low false negatives). Finally, the Average Loss (0.0068) indicates that the predictions made by the model will be consistently close to the target values predicted by the loss function that was traditionally used, and further reinforces the strong quantitative performance across the board for this crack segmentation application.

Table 3.2 presents a quantitative comparison of the proposed UNet++ model against Modified U-Net model on a semantic segmentation task, for crack detection over 1579 test sample.

Table 3.2: Performance Evaluation of Segmentation Proposed Models.

Proposed Model	**Dice score**	**IoU score**	**F1 score**	**Precision**	**Recall**	**Accuracy**
UNet++	0.9338	0.8805	0.9609	0.9603	0.9614	99.71%
U-Net	0.8706	0.7802	0.9085	0.9287	0.8892	99.33%

Figure 3.8 shows a visual comparison between input images, ground truth crack masks, and predicted crack masks (with a threshold of 0.5) from a trained segmentation model.

The predicted masks also show a high similarity to the real-world masks in terms of crack shape, orientation, and continuity. The segmentation appears accurate and consistent, even in the presence of complex texture and background noise. This confirms the model's strong generalizability and accurate performance in detecting crack boundaries in images.

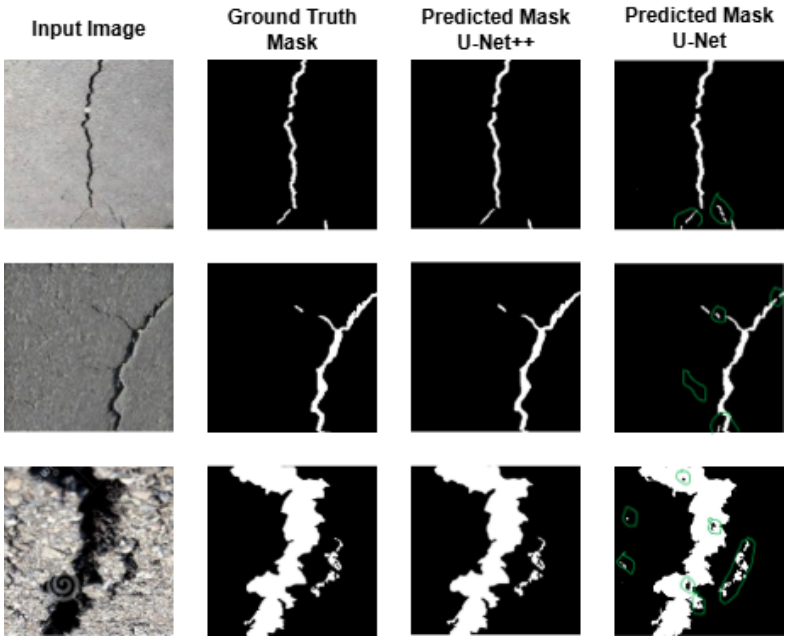

Figure 3.8: Qualitative Evaluation of Crack Segmentation Results.

3.8 Comparison of our method with previous methods

Table 3.3 provides a quantitative comparison of the proposed UNet++ model's performance against several other established deep learning-based crack detection methods, using standard evaluation metrics. The results clearly demonstrate the superiority of the proposed UNet++ model, which achieved the highest F1-score (96.09%), Precision (96.03%), Recall (96.14%), and IoU score (88.05%). This performance significantly surpasses that of other notable methods, including DeepCrack [6] (IoU 85.9%, F1-score 86.5%), DDLCN [69] (F1-score 90.98%). The consistent lead across these critical metrics underscores the effectiveness of the UNet++ architecture and the implemented training strategies for robust and accurate crack segmentation compared to contemporary approaches.

Table 3.3: Comparative performance of crack detection methods using standard evaluation metrics.

Method	**F1 score**	**Precision**	**Recall**	**IoU score**	**Inference Time per Image**	**Throughput**
DeepCrack [6]	86.5	86.8	86.9	85.9	0.1 sec	10 images/sec
Efficient UNet [40]	67.3	82.39	56.88	-	-	-
Hybrid-Segmentor [39]	77	80.4	74.4	63	$\tilde{0}$.0286 sec	35 images/sec
DDLCN [66]	86.09	88.80	83.09	-	-	-
Our method	96.09	96.03	96.14	88.05	0.0009 sec	1077.10 images/sec

3.9 Development of a simple crack detection system

This crack detection system allows users to interact with a multi-component application, which distinguishes between a user-facing front-end and a processing back-end. Users can either load a static image or utilize a live camera feed; for phone camera input, this is typically achieved by running a separate app on the phone "IP Webcam" that broadcasts its camera feed over the local Wi-Fi network, which the desktop application's back-end then connects to via a specific network URL using OpenCV. The front-end, developed with PyQt5, provides the graphical user interface (GUI) for these input methods and displays the processed results. The back-end manages the core logic: after image acquisition, a data processing pipeline prepares images (resizing, normalization via torchvision transforms) for the inference stage. The heart of the system is a pre-trained UNetPlusPlus deep learning model, implemented in PyTorch, which performs the semantic segmentation to identify crack pixels, processes this incoming visual data. This AI model analyzes each image or video frame to produce a probability map highlighting potential crack regions, which is then refined into a distinct mask, resized, and overlaid onto the original visual, highlighting detected cracks in real-time (for camera input) or on the static image.

We tested the system by inputting different images containing cracks. We obtained the results shown in Figure 3.9, which is a mask corresponding to the input image, as well as an image in which the location of the crack is indicated in red. When using a live feed from a phone camera of a house wall and then the "Ahbas" dam located in Ghardaia province, cracks were detected in real time (after the camera was input), and the results are shown in this *Link to Google Drive File*.

Figure 3.10 visually highlights the broad applicability of the proposed crack detection system, demonstrating its usefulness in monitoring various critical infrastructures, such as monitoring long-distance bridges over water, assessing the structural integrity of large concrete dams, inspecting urban building facades and structures, and assessing the condition of highway surfaces. Together, these images demonstrate the versatile system design, capable of identifying defects and supporting structural integrity monitoring across a variety of critical civil engineering assets.

3.10 Limitations and Future Directions

Although the results are encouraging, the model was only tried and tested mainly on the DeepCrack dataset. Future work must include testing, and possibly

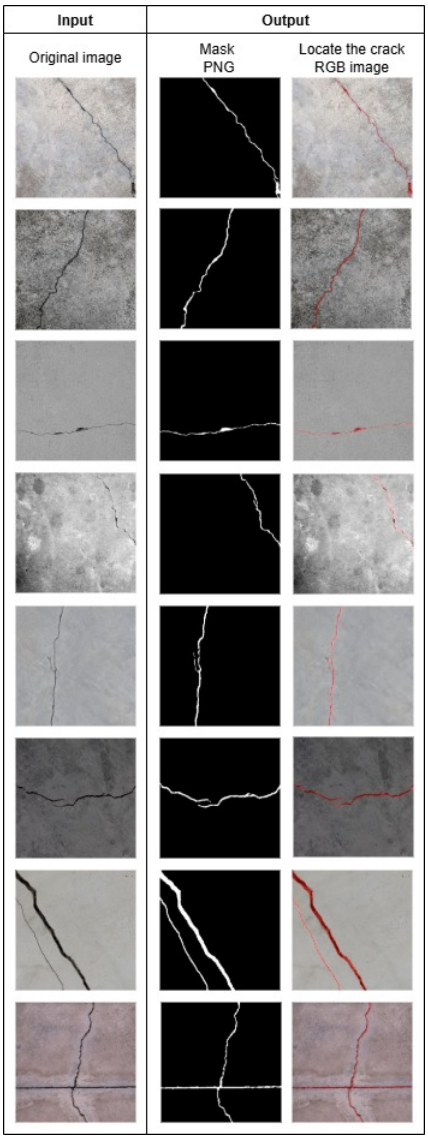

Figure 3.9: Visual results of the crack detection system.

Figure 3.10: The described crack detection system targets a wide range of civil infrastructure, highlighting its versatile applicability for structural health monitoring.

additional refinement, of the model on an even broader range of datasets across other surface materials (e.g., metal, other composites), crack types, and imaging conditions (e.g., different lighting, shadows, occlusions) to further evaluate its robustness and usability. The prototype system is capable of operating in real-time, but can be optimized more for deployment on resource-constrained edge devices used during field inspections, for example, surveillance cameras or drones. It would be ideal to pursue different modes of model compression and acceleration with minimal performance loss that is allowed by the end user. Moreover, extending the system to also quantify crack characteristics (e.g., length, width, density, orientation) and automatically classify the severity of cracks would be a significant contribution. Finally, with respect to reducing the dependence on fully annotated large datasets, it would be advantageous to explore unsupervised or semi-supervised learning options that lessen the need for meticulously annotated datasets, which are often costly and time-consuming to assemble.

Conclusion

The chapter effectively demonstrated the viability and performance of the proposed deep learning model for crack detection from a series of systematic experiments. Given a prepared dataset and good data augmentation and segmentation, the proposed model achieved very good performance metrics (Dice score, IoU, precision, recall, F1-score). In comparison to other methods of previous research, the proposed model was often very impressive in its performance. Moreover, seeing cracks detected in real-time and using a pre-trained UNet++ model has demonstrated the feasibility and implications for civil infrastructure applications. All in all, the proposed system provides a noteworthy contribution to the area of scalable, accurate and lightweight systems for the identification of cracks.

General Conclusion

This thesis presented a comprehensive study of crack detection based on deep learning and posited semantic segmentation as a superior method to the automatic extraction of structural surface defects. The work began with the disadvantages of traditional inspection methods and, having identified those shortcomings, moved on to rationalizing a move towards deep learning. This was preceded by a literature review of the most prevalent architectures and techniques in the field aimed at this task, which then informed the selection of the most appropriate model.

Next, a cutting-edge encoder-decoder architecture, UNet++, was utilized to provide a solution for precise crack detection. The model was trained and validated using the public DeepCrack dataset, which was preprocessed and extensively augmented in an organized manner for improved generalization. Common measures were employed to evaluate the model's performance and output and compared with alternative solutions. The model performed better in segmentation accuracy and robustness and therefore has potential for deployment in real-time detection.

Additionally, a reduced complexity detection system was also developed to demonstrate applicability. The results, while promising, did not come without facets of ongoing issues such as data annotation bottleneck, real-world variability and computational efficiency.

In conclusion, this study showed the benefits of deep learning and semantic segmentation as a tool for infrastructure inspection. It provides a basis for future research towards fully automated, scalable, and intelligent crack monitoring systems that elevate the safety and sustainability of civil structures.

Bibliography

[1] *R. Hoensheid, "Evaluation of surface defect detection in reinforced concrete bridge decks using terrestrial lidar," Master's thesis, University of Missouri–Columbia, 2012.*

[2] *D. Shojaei, P. Jafary, and Z. Zhang, "Mixed Reality-Based Concrete Crack Detection and Skeleton Extraction Using Deep Learning and Image Processing,"* Electronics, *vol. 13, no. 22, p. 4426, Nov. 2024. [Online]. Available: magentahttps://doi.org/10.3390/electronics13224426*

[3] *Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," in* Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support (DLMIA/ML-CDS 2018)*, ser. Lecture Notes in Computer Science, vol. 11045. Springer, 2018, pp. 3–11. [Online]. Available: magentahttps://doi.org/10.1007/978-3-030-00889-5_1*

[4] *V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation,"* IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.*

[5] *S. Choi and M. Lee, "Transformer Architecture and Attention Mechanisms in Genome Data Analysis: A Comprehensive Review,"* Biology*, vol. 12, no. 7, p. 1033, Jul. 2023. [Online]. Available: magentahttps://doi.org/10.3390/biology12071033*

[6] *Y. Liu, J. Yao, X. Lu, R. Xie, and L. Li, "Deepcrack: A deep hierarchical feature learning architecture for crack segmentation,"* Neurocomputing*, vol. 338, pp. 139–153, 2019.*

[7] *J. Canny, "A computational approach to edge detection,"* IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.*

[8] *N. Otsu, "A threshold selection method from gray-level histograms,"* IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.*

[9] *P. Subirats, J. Dumoulin, V. Legeay, and D. Barba, "Automation of pavement surface crack detection using the continuous wavelet transform," in* 2006 International Conference on Image Processing*, Oct. 2006, pp. 3037–3040.*

[10] *Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang, "Cracktree: Automatic crack detection from pavement images,"* Pattern Recognition Letters*, vol. 33, no. 3, pp. 227–238, 2012.*

[11] *H. Cheng, J.-R. Chen, C. Glazier, and Y. Hu, "Novel Approach to Pavement Cracking Detection Based on Fuzzy Set Theory,"* Journal of Computing in Civil Engineering*, vol. 13, no. 4, pp. 270–280, Oct. 1999. [Online]. Available: magentahttps://doi.org/10.1061/(ASCE)0887-3801(1999)13:4(270)*

[12] *C. Cortes and V. Vapnik, "Support-vector networks,"* Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.*

[13] *H.-G. Moon and J.-H. Kim, "Intelligent crack detecting algorithm on the concrete crack image using neural network," in* Proceedings of the 28th International Symposium on Automation and Robotics in Construction (ISARC)*, 2011.*

[14] *R.-S. Rajadurai and S.-T. Kang, "Automated vision-based crack detection on concrete surfaces using deep learning,"* Applied Sciences*, vol. 11, no. 11, p. 5229, 2021.*

[15] *A. Sarhadi, M. Ravanshadnia, A. Monirabbasi, and M. Ghanbari, "Optimizing concrete crack detection: An attention-based swin u-net approach,"* IEEE Access*, vol. 12, pp. 77575–77585, 2024.*

[16] *Y.-A. Hsieh and Y. J. Tsai, "Machine learning for crack detection: Review and model performance comparison,"* Journal of Computing in Civil Engineering*, vol. 34, no. 5, p. 04020038, 2020.*

[17] *S. P. Timoshenko and J. N. Goodier,* Theory of Elasticity*, 3rd ed. New York: McGraw-Hill, 1970.*

[18] *P. K. Mehta and P. J. Monteiro,* Concrete: Microstructure, Properties, and Materials*, 4th ed. McGraw-Hill Education, 2014.*

[19] *C. R. Farrar and K. Worden,* Structural Health Monitoring: A Machine Learning Perspective. *John Wiley & Sons, 2012.*

[20] *D. L. Balageas, C. P. Fritzen, and A. Güemes, Eds.,* Structural Health Monitoring. *ISTE Ltd., 2006.*

[21] *Y. Chen, Z. Zhu, Z. Lin, and Y. Zhou, "Building surface crack detection using deep learning technology,"* Buildings*, vol. 13, no. 7, p. 1814, 2023.*

[22] *W. Choi, "Deep learning implemented structural defect detection on digital images," Doctoral dissertation, University of Manitoba, 2020.*

[23] *Y. Maode, B. Shaobo, X. Kun, and H. Yuyao, "Pavement crack detection and analysis for high-grade highway," in* The Eighth International Conference on Electronic Measurement and Instruments*, 2007, pp. 4–548–4–552.*

[24] *H. Li, J. Zong, J. Nie, Z. Wu, and H. Han, "Pavement crack detection algorithm based on densely connected and deeply supervised network,"* IEEE Access*, vol. 9, pp. 11 835–11 842, 2021.*

[25] *B. R. Ellingwood, "Risk-informed condition assessment of civil infrastructure: state of practice and research issues,"* Structure and Infrastructure Engineering*, vol. 1, no. 1, pp. 7–18, 2005.*

[26] *B. M. Phares, D. D. Rolander, B. A. Graybeal, and G. A. Washer, "Reliability of visual bridge inspection (fhwa-rd-01-020)," Federal Highway Administration, Tech. Rep., 2001.*

[27] *P. Drukis, L. Gaile, and L. Pakrastins, "Inspection of public buildings based on risk assessment,"* Procedia Engineering*, vol. 172, pp. 247–255, 2017.*

[28] *Y. Cha, W. Choi, and O. Büyüköztürk, "Deep learning-based crack damage detection using convolutional neural networks,"* Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, pp. 361–378, 2017.*

[29] *J. Tang and Y. Gu, "Automatic crack detection and segmentation using a hybrid algorithm for road distress analysis," in* 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC). *IEEE, 2013, pp. 3026–3030.*

[30] *A. Ayenu-Prah and N. Attoh-Okine, "Evaluating pavement cracks with bidimensional empirical mode decomposition,"* EURASIP Journal on Advances in Signal Processing*, p. 861701, 2008.*

[31] K. Fernandes and L. Ciobanu, "Pavement pathologies classification using graph-based features," in 2014 IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 793–797.*

[32] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in 2016 IEEE International Conference on Image Processing (ICIP)*, Phoenix, AZ, USA, 2016, pp. 3708–3712.*

[33] A. Müller, N. Karathanasopoulos, C. C. Roth, and D. Mohr, "Machine learning classifiers for surface crack detection in fracture experiments," International Journal of Mechanical Sciences*, vol. 209, p. 106698, 2021.*

[34] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature*, vol. 323, no. 6088, pp. 533–536, 1986.*

[35] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning*. MIT press, 2016.*

[36] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature*, vol. 521, no. 7553, pp. 436–444, 2015.*

[37] Y. Hamishebahar, H. Guan, S. So, and J. Jo, "A comprehensive review of deep learning-based crack detection approaches," Applied Sciences*, vol. 12, no. 3, p. 1374, 2022.*

[38] Z. Tong, T. Ma, and J. Huyan, "Pavementscapes: a large-scale hierarchical image dataset for asphalt pavement damage segmentation," arXiv preprint arXiv:2208.00775*, 2022.*

[39] J. M. Goo, X. Milidonis, A. Artusi, J. Boehm, and C. Ciliberto, "Hybrid-segmentor: A hybrid approach to automated fine-grained crack segmentation in civil infrastructure," arXiv preprint arXiv:2409.02866*, 2024.*

[40] W. Wang and C. Su, "Semi-supervised semantic segmentation network for surface crack detection," Automation in Construction*, vol. 128, p. 103786, 2021.*

[41] S. Egodawela, A. Khodadadian Gostar, H. A. D. S. Buddika, A. J. Dammika, N. Harischandra, S. Navaratnam, and M. Mahmoodian, "A deep learning approach for surface crack classification and segmentation in unmanned aerial vehicle assisted infrastructure inspections," Sensors*, vol. 24, no. 6, p. 1936, 2024.*

[42] *W. R. L. da Silva and D. S. de Lucena, "Concrete cracks detection based on deep learning image classification," in* Proceedings 2018*, vol. 2, no. 8, 2018, p. 489.*

[43] *Y. Song, W. Xia, Y. Li, H. Li, M. Yuan, and Q. Zhang, "Anomalyseg: Deep learning-based fast anomaly segmentation approach for surface defect detection,"* Electronics*, vol. 13, no. 2, p. 284, 2024.*

[44] *L. Wang, L. Zhuang, and Z. Zhang, "Automatic detection of rail surface cracks with a superpixel-based data-driven framework,"* Journal of Computing in Civil Engineering*, vol. 33, no. 1, p. 04018053, 2019.*

[45] *F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, and H. Ling, "Feature pyramid and hierarchical boosting network for pavement crack detection,"* IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1525–1535, 2019.*

[46] *X. Dong, Y. Liu, and J. Dai, "Concrete surface crack detection algorithm based on improved yolov8,"* Sensors*, vol. 24, no. 16, p. 5252, 2024.*

[47] *X. Zheng, S. Zhang, X. Li, G. Li, and X. Li, "Lightweight bridge crack detection method based on segnet and bottleneck depth-separable convolution with residuals,"* IEEE Access*, vol. 9, pp. 161 649–161 669, 2021.*

[48] *A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks,"* Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.*

[49] *Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition,"* Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.*

[50] *S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks,"* arXiv preprint arXiv:1506.01497*, 2015.*

[51] *J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.*

[52] *Z. Yu, "Yolo v5s-based deep learning approach for concrete cracks detection," in* SHS Web of Conferences*, vol. 144, no. 5, 2022, p. 03015.*

[53] *J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in* Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.*

[54] *O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in* International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 234–241.*

[55] *T. Chisholm, R. G. Lins, and S. N. Givigi, "Fpga-based design for real-time crack detection based on particle filter," in* 2018 IEEE International Symposium on Industrial Electronics (ISIE)*, 2018, pp. 101–106.*

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

[57] *K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in* Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, 2016, pp. 770–778.*

[58] *D. M. Jenkins, T. A. Carr, M. I. Iglesias, T. Buggy, and G. Morison, "A deep convolutional neural network for semantic pixel-wise segmentation of road and pavement surface cracks," in* 2018 26th European Signal Processing Conference (EUSIPCO)*, 2018, pp. 2120–2124.*

[59] *L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," in* Computer Vision -- ECCV 2018*, ser. Lecture Notes in Computer Science, vol. 11211. Springer International Publishing, 2018, pp. 833–851. [Online]. Available: magentahttps://doi.org/10.1007/978-3-030-01234-2_49*

[60] *G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in* Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 2261–2269.*

[61] *S. Jegou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 11–19.*

[62] *A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly* et al.*,*

"An image is worth 16x16 words: Transformers for image recognition at scale," arXiv preprint arXiv:2010.11929*, 2020.*

[63] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo et al.*, "Segment anything,"* arXiv preprint arXiv:2304.02643*, 2023.*

[64] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.*

[65] H. Fu, D. Meng, W. Li, and Y. Wang, "Bridge crack semantic segmentation based on improved Deeplabv3+," Journal of Marine Science and Engineering*, vol. 9, no. 6, p. 671, 2021.*

[66] J. Cheng, W. Xiong, W. Chen, Y. Gu, and Y. Li, "Pixel-level crack detection using U-Net," in TENCON 2018 - 2018 IEEE Region 10 Conference*. IEEE, 2018, pp. 0462–0466.*

[67] K. Gao, "U-Net based crack detection in road and railroad tunnels using data acquired by mobile device," Master's Thesis, Second Cycle, 30 Credits, KTH Royal Institute of Technology, School of Electrical Engineering and Computer Science, Stockholm, Sweden, 2022, uRN: TRITA-ABE-MBT-2248 (or similar, check DiVA portal).

[68] D. Tabernik, Šela, J. Skvarč, and D. Skočaj, "Segmentation-based deep-learning approach for surface-defect detection," Journal of Intelligent Manufacturing*, vol. 31, no. 3, pp. 759–776, Mar. 2019.*

[69] L. Fan and J. Zou, "A novel road crack detection technology based on deep dictionary learning and encoding networks," Applied Sciences*, vol. 13, no. 22, p. 12299, 2023.*

[70] B. Li, X. Chu, F. Lin, F. Wu, S. Jin, and K. Zhang, "A highly efficient tunnel lining crack detection model based on Mini-Unet," Scientific Reports*, vol. 14, no. 1, p. 28234, 2024.*

[71] P. Siva, D. Yamaganti, D. Rohita, and U. Sikharam, A Review on Python for Data Science, Machine Learning and IOT*, November 2023.*

[72] K. Ma, X. Meng, M. Hao, G. Huang, Q. Hu, and P. He, "Research on the efficiency of bridge crack detection by coupling deep learning frameworks with convolutional neural networks," Sensors*, vol. 23, no. 16, p. 7272, 2023.*

[73] *M. M. M. Fareez, V. Thangarajah, and S. Saabith, "Popular python libraries and their application domains," Nov. 2020.*

[74] *D. K. Sydykova, B. R. Jack, S. J. Spielman, and C. O. Wilke, "Measuring evolutionary rates of proteins in a structural context,"* F1000Research*, vol. 6, p. 1845, 2018, version 2; peer review: 4 approved. [Online]. Available: magentahttps://doi.org/10.12688/f1000research.12874.2*

[75] *X. Chen, "Introduction and analysis of python software,"* Frontiers in Computing and Intelligent Systems*, vol. 5, pp. 41–43, 2023.*

[76] *P. S. Foundation. (2025) sys — System-specific parameters and functions. magentahttps://docs.python.org/3/library/sys.html. Accessed: 2025-05-24.*

[77] *DataCamp. (2024, September) Tqdm python: A guide with practical examples. Accessed: 2025-05-24. [Online]. Available: magentahttps://www.datacamp.com/tutorial/tqdm-python*

[78] *T. Boyle. (2025, March) Glob module in python: Explained. Updated by Brennan Whitfield. [Online]. Available: magentahttps://builtin.com/software-engineering-perspectives/glob-in-python*

[79] *A. Miller, N. Kobylski, E. Qamar, J. Xiao, N. Veal, R. Kenney, N. Wysocki, and M. Mahmoud,* Automating File Operations via Python*, December 2022.*

[80] *C. L. Angelina and A. Rospawan, "A benchmark study of deeplabv3+, u-net++, and attention u-net for blood cell segmentation,"* Green Intelligent Systems and Applications*, vol. 5, no. 1, pp. 61–73, 2025. [Online]. Available: magentahttps://doi.org/10.53623/gisa.v5i1.607*

République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de La Recherche Scientifique
كلية العلوم والتكنولوجيا
Faculté des Sciences et de la Technologie
قسم الرياضيات و الإعلام الآلي
Département des Mathématiques & de l'Informatique
جامعة غرداية
Université de Ghardaia

شهادة الترخيص بالإيداع

أنا الأستاذ : سليمان بالاعور

بصفتي رئيس لجنة المناقشة والمسؤول عن تصحيح مذكرة الماستر الموسومة ب:

Deep Learning-based Surface Crack Detection

من إنجاز الطالب: **لامين محمد الأمين**

والطالب: **نجار صلاح الدين**

الكلية: العلوم والتكنولوجيا
القسم: الرياضيات والإعلام الآلي
الشعبة: إعلام آلي

التخصص: الأنظمة الذكية لاستخراج المعارف
تاريخ المناقشة: 2025/07/02

أشهد أن الطلبة قاموا بالتصحيحات المطلوبة من طرف لجنة المناقشة وأن المطابقة بين النسخة الورقية والالكترونية استوفيت جميع شروطها.

إمضاء المسؤول عن التصحيح

مصادقة رئيس القسم

رئيس قسم الرياضيات والإعلام الآلي
الحاج موسى ياسين

جامعة غرداية
قسم الرياضيات والإعلام الآلي