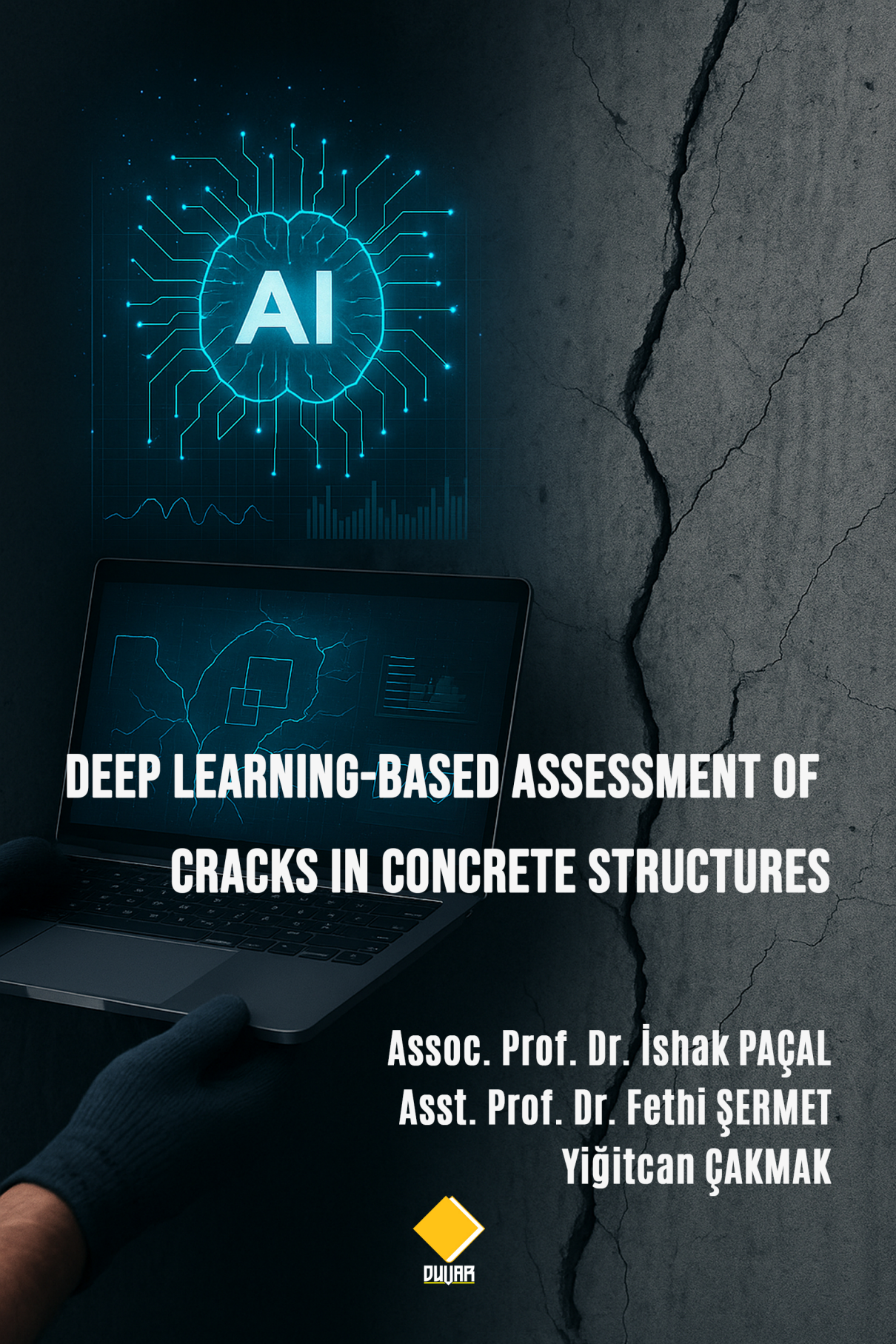

DEEP LEARNING-BASED ASSESSMENT OF CRACKS IN CONCRETE STRUCTURES

Assoc. Prof. Dr. İshak PAÇAL
Asst. Prof. Dr. Fethi ŞERMET
Yiğitcan ÇAKMAK

DUVAR

DEEP LEARNING-BASED ASSESSMENT OF CRACKS IN CONCRETE STRUCTURES

Assoc. Prof. Dr. İshak PAÇAL

Asst. Prof. Dr. Fethi ŞERMET

Yiğitcan ÇAKMAK

Deep Learning-Based Assessment of Cracks in Concrete Structures

Assoc. Prof. Dr. İshak PAÇAL, Asst. Prof. Dr. Fethi ŞERMET

Yiğitcan ÇAKMAK

Editor in chief: Berkan Balpetek

Printing : JULY -2025

Publisher Certificate No: 49837

ISBN: 978-625-5885-67-8

853 Sokak No:13 P.10 Kemeraltı-Konak/İzmir

Tel: 0 232 484 88 68

www.duvaryayinlari.com

duvarkitabevi@gmail.com

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abstract

Ensuring the structural integrity of concrete infrastructure, a cornerstone of modern civilization, necessitates the timely and accurate detection of cracks. Traditional visual inspection methods, however, are fraught with limitations, including subjectivity, high costs, and significant labor investment. To address these deficiencies, this investigation presents one of the most extensive comparative analyses to date, evaluating the performance of dozens of state-of-the-art deep learning architectures for automated crack detection. Spanning seven distinct model families including seminal Convolutional Neural Networks (ResNets, DenseNets, EfficientNet-V2) and paradigm-shifting Transformers (ViT, Swin, BEiT) the models were rigorously tested on the diverse SDNET2018 dataset. The methodology encompassed a dual-phase experimental design: first, three independent binary classification tasks for Deck, Pavement, and Wall surfaces, followed by a more demanding six-class classification task on a unified dataset to assess both defect detection and contextual identification capabilities. All architectures were fine-tuned using a standardized training protocol on a class-balanced dataset to ensure a fair and robust comparison. The experimental findings reveal a clear performance hierarchy, with CNN-based architectures, particularly the ResNet and EfficientNet-V2 families, demonstrating more consistent and superior efficacy than their Transformer-based counterparts across most tested scenarios. A central and compelling discovery from this study is the remarkable performance of EfficientNet-V2 Small, a compact model that not only competed with but frequently surpassed much larger architectures, achieving the highest F1-score in both the 'Wall' and the complex six-class classification tasks. Similarly, the classic ResNet-34 architecture proved its enduring relevance by emerging as the top performer on the visually noisy 'Pavement' dataset. Among the Transformer models, Swin Transformer, which reincorporates principles of hierarchy and locality, exhibited the most competitive performance, whereas standard ViT and BEiT models yielded more modest results. Ultimately, this research robustly demonstrates that for practical engineering applications like

automated crack detection, computational efficiency is not a trade-off against performance but rather a potential catalyst for it. The evidence suggests that compact, well-designed CNNs such as EfficientNet-V2 Small and ResNet-18/34 provide an optimal balance between high accuracy, low computational overhead, and rapid inference capabilities. These findings furnish a definitive, data-driven roadmap for the development and deployment of reliable, real-time automated inspection systems on resource-constrained platforms, including mobile devices and unmanned aerial vehicles.

INTRODUCTION

Concrete, the cornerstone of modern civilization, stands out as an indispensable material in the construction of critical infrastructure systems such as bridges, buildings, dams, and highways. Its superior properties, such as high compressive strength, cost-effectiveness, and ease of on-site production, have made it the most widely used construction material on a global scale [1,2]. However, despite this widespread use of concrete, it is inevitable for it to degrade over time as a result of environmental factors, mechanical loads, and chemical interactions [3,4]. The most prominent and early harbinger of these degradation processes is the cracks that occur on the surface and in the internal structure. These cracks, which are initially at a micro-level, can grow over time, if necessary, precautions are not taken, becoming a serious threat to the structure's load-bearing capacity, durability, and service life [5].

In the field of Structural Health Monitoring (SHM), traditional methods used for the detection of concrete cracks are largely based on human observation and expert experience. Periodic visual inspections conducted by field engineers or inspectors form the basis of this process [6]. However, this traditional approach has many significant disadvantages. Primarily, the inspection process is extremely time-consuming, labor-intensive, and costly. Second, the accuracy and consistency of the detection are directly dependent on subjective and variable factors such as the inspector's fatigue, attention, experience, and lighting conditions. Furthermore, inspections in high or hard-to-reach areas pose serious safety risks for personnel. These limitations reduce the efficiency of traditional methods and clearly reveal the need for more objective, fast, and reliable alternatives [7].

The inadequacies of traditional methods have directed researchers towards computer vision and artificial intelligence technologies [8]. Although initial image processing-based approaches used algorithms such as edge detection, thresholding, and morphological operations, they could not produce robust results in the face of the complexity of real-world field conditions (lighting changes,

shadows, moisture stains, surface pollution). At this point, deep learning, and particularly Convolutional Neural Networks (CNNs), has created a revolution in image-based recognition tasks [9]. Unlike traditional machine learning (ML) [10,11] approaches, CNNs possess the ability to automatically learn hierarchical and meaningful features from raw pixel data [12,13]. Thereby, they have offered much more effective and generalizable solutions to the crack detection problem by eliminating the need for manual feature engineering [14,15]. These achievements have made CNN-based architectures indispensable in the medical field; to give some examples of areas where they are used, brain tumors [16–18], breast cancer [19], lung cancer [20], dentistry [21,22], and urology [23] are just a small fraction. With developing new technologies, artificial intelligence continues to demonstrate its success in every field.

The success of CNN-based approaches has paved the way for the development of progressively deeper and more complex architectures. Within this sphere, the ResNet (18, 34, 50, 101, 152) family has become an industry standard by using residual connections to enable the training of very deep networks. The DenseNet (121, 169, 201) architecture has provided high parameter efficiency by maximizing feature propagation and reuse through dense connections that link each layer to all subsequent layers. Following these two foundational architectures, the EfficientNetV2 (small, medium, large) family emerged, which systematically scales the model's depth, width, and resolution. This model offered the possibility of achieving high performance even with limited resources by establishing an optimized balance between accuracy and computational efficiency.

The latest breakthrough in the field of computer vision has been the adaptation of the Transformer architecture, which revolutionized the field of natural language processing (NLP), to vision. The Vision Transformer (ViT) (with patch 16/32 variants of tiny, small, base, large), which processes an image as a sequence of "patches" and models the global relationships between these patches with a self-attention mechanism, has brought a new perspective by overcoming the local

receptive field limitation inherent to CNNs. The Swin Transformer (tiny, small, base, large), which increases computational efficiency by combining this approach with a hierarchical structure and shifted windows, and BEiT (base, large), which learns robust representations with less labeled data through self-supervised pre-training, stand out as the most advanced representatives of this new paradigm.

As the detection of cracks in concrete structures is a critical task for ensuring structural integrity and safety, the research community has focused on overcoming the inherent limitations of traditional visual inspection methods, such as being time-consuming, labor-intensive, and subjective. The consensus in the literature is that automation is imperative to address these challenges (Kirthiga and Elavenil; Panwar et al.) [24,25]. Accordingly, systems based on ML and particularly Deep Learning (DL) have emerged as the dominant paradigm due to their potential for high accuracy, efficiency, and reliability. A comprehensive review by Pandey and Mishra [26] illustrates the technological evolution in this domain, comparing a wide array of approaches from classical ML algorithms like Random Forest (RF) to more advanced DL architectures like CNNs. Similarly, Arpitha et al. [27], after reviewing works from 1999 to 2023, confirm that ML and DL-based methods are increasingly preferred due to their advantages in automation and precision.

In this pursuit of automation, CNNs have become the de facto standard for image-based crack detection tasks. The ability of CNNs to automatically learn hierarchical and discriminative features from raw pixel data makes them significantly superior to traditional ML methods. A comparative analysis by Navpreet et al. [28] concretely demonstrates this superiority, showing that a pre-trained CNN model like VGG16, with an accuracy of 92.14%, vastly outperforms the best-performing traditional ML classifier, Random Forest (66.92%). The review by Pandey and Mishra corroborates this finding, noting that VGG-16 can achieve exceptional accuracy rates as high as 99.83%. Studies such as Usha's [29] "DeepCrack" and the novel deep CNN model developed by Abbas and Alghamdi

[30] prove the potential of these architectures to create specialized, high-performance solutions capable of successfully detecting both visible and subtle cracks under varying lighting and surface texture conditions.

Research has extended beyond foundational CNN architectures to encompass more sophisticated approaches and practical application scenarios. One of the most significant advancements in this area is the adoption of transfer learning, a technique that leverages the knowledge from models pre-trained on large datasets and applies it to smaller, specific datasets. The work by Bussa and Boppana [31] exemplifies this, achieving a 97% F1-score using the ResNet50 architecture with transfer learning on the METU dataset and demonstrating its superiority over VGG-based models. Similarly, Dai et al. [32] utilized modern architectures like ResNet50 and EfficientNetB1 with transfer learning to detect cracks in dams, achieving the highest performance with a hybrid model. In addition to these theoretical advancements, practical applications, such as the work by Wang et al. [33], expand the field's practical potential by integrating technologies like Unmanned Aerial Vehicles (UAVs) to automate the data collection process and visualizing the results on 3D models.

A comprehensive analysis of the existing literature unequivocally establishes the effectiveness and superiority of deep learning-based methods, especially advanced CNN architectures, for concrete crack detection. The use of models like ResNet and EfficientNet with transfer learning is recognized as a robust approach that delivers high accuracy rates. However, most of these studies tend to focus on a single architectural family or compare a limited number of models. A noticeable gap exists in the literature for a study that systematically and comprehensively compares established CNN architectures like ResNet and DenseNet, the latest generation of efficiency-focused CNNs like EfficientNetV2, and models from an entirely different paradigm namely Transformers such as ViT, Swin Transformer, and BEiT on the same controlled dataset, at a wide scale (e.g., from "tiny" to "large"), and analyzes them in terms of performance, efficiency, and

generalization capabilities. This study is designed to fill this precise methodological gap.

In light of the aforementioned architectural advancements, the central objective of this study is to present a comprehensive and comparative DL analysis for the detection and classification of cracks on concrete surfaces, utilizing the large-scale, publicly available SDNET2018 dataset. This work seeks to provide an original contribution to the literature through the systematic evaluation of a diverse array of models representing distinct architectural paradigms. Accordingly, the investigation involves the training and comparative assessment of selected models from CNN-based families, including ResNet, DenseNet, and EfficientNetV2, as well as from Transformer-based families such as ViT, Swin Transformer, and BEiT. The performance of these models is quantitatively appraised using standard metrics, encompassing accuracy, precision, recall, and the F1-score. The resulting analysis serves to elucidate the relative performance of these different architectures in the crack detection task. Furthermore, the findings are intended to furnish a practical guide for real-world applications and to inform the direction of subsequent research in this domain.

MATERIALS AND METHODS

Dataset

The experimental foundation of this investigation is built upon the SDNET2018 dataset, a large-scale and publicly accessible collection that is widely regarded as a cornerstone in the field. Its selection was predicated not merely on the substantial number of images it contains, but on its inherent complexity and diversity, which mirror real-world conditions. As a comprehensive benchmark, this collection comprises over 56,000 high-resolution images meticulously labeled for multi-class crack detection and classification across three primary structural categories: concrete pavements, structural walls, and bridge decks [34,35]. This composition elevates the problem beyond a simple binary (crack/non-crack) decision, presenting a more formidable challenge that requires a model to concurrently identify both the type of structural surface and the presence of a defect. Figure 1 presents a selection of class-based examples randomly drawn from the SDNET2018 dataset, showcasing the richness and variety in terms of texture, lighting conditions, and crack morphologies.

Figure 1. Sample images from the dataset for cracked and non-cracked classes across different concrete structure types (Decks, Pavements, and Walls).

The dataset is structured into six primary classes, comprising 'cracked' and 'non-cracked' instances for each of the three structural categories. In its raw form, however, the SDNET2018 dataset presents a significant challenge for training DL models due to a pronounced class imbalance. The original distribution contains a substantially larger number of non-cracked images compared to cracked images within each category. Such an imbalance can induce a bias in the model during training, causing it to favor the majority (non-cracked) class. A model trained under these conditions would tend to predict non-cracked surfaces with high accuracy while potentially overlooking or misclassifying the critical minority (cracked) class. To mitigate this potential bias and to provide a fair learning environment for the models, a deliberate balancing strategy was implemented during the data pre-processing stage. This strategy involved random down sampling, where samples from the more populous non-cracked class were randomly discarded to match the number of images in the cracked class for each category. As a result of this process, the original dataset of over 56,000 images was reduced to a more manageable and balanced size of 16,968 images. To further deepen the comparative analysis and to assess the models' capabilities beyond defect detection to include contextual classification, an additional experimental phase was designed. In this phase, the previously separate 'Deck', 'Pavement', and 'Wall' categories were consolidated into a single, unified dataset with their respective 'cracked' and 'non-cracked' labels. This consolidation resulted in a new, more challenging multi-class classification problem with six distinct classes. This task requires the models not only to determine the presence of a crack but also to simultaneously distinguish the type of structural surface on which it appears. To maintain consistency, this unified dataset was also partitioned into training, validation, and test sets with a 70%, 15%, and 15% split, respectively. All architectures employed in the study were subsequently fine-tuned for this six-class task using a transfer learning approach. This methodology facilitates a multi-faceted comparison, testing both the defect detection sensitivity and the discriminative power of each architecture across environments with different visual textures and patterns. A detailed distribution of the dataset is provided in Table 1.

Table 1. Detailed distribution of the image dataset across structural types, classes, and data splits (Training, Validation, Test).

Class	Train		Validation		Test		Total
	Cracked	Non-cracked	Cracked	Non-cracked	Cracked	Non-cracked	
Deck	1417	1417	303	303	305	305	4050
Pavement	1825	1825	391	391	392	392	5216
Wall	2695	2695	577	577	579	579	7702
Total	11874		2542		2552		16968

To facilitate an objective evaluation of model performance, this final, balanced dataset was partitioned into three fundamental subsets in accordance with standard ML protocols: 70% for training, 15% for validation, and 15% for testing. The training set is designated for the model to learn the underlying patterns and features from the data, while the validation set serves as a feedback mechanism during the training process to optimize hyperparameters and prevent overfitting. The test set, which consists of data entirely unseen during the training phase, provides the final and unbiased measure of a model's ability to generalize. Table 1 offers a granular and numerical breakdown of these 16,968 images, detailing their distribution across the three distinct structural domains and the aforementioned subsets. An examination of the table reveals the differences in the total number of samples among the categories: Bridge Decks (4,050 images), Pavements (5,216 images), and Walls (7,702 images). Complementing this numerical account, Figure 2 presents the same distribution in a visual format, facilitating an intuitive understanding of the proportions within each structural category for the training, validation, and test sets.

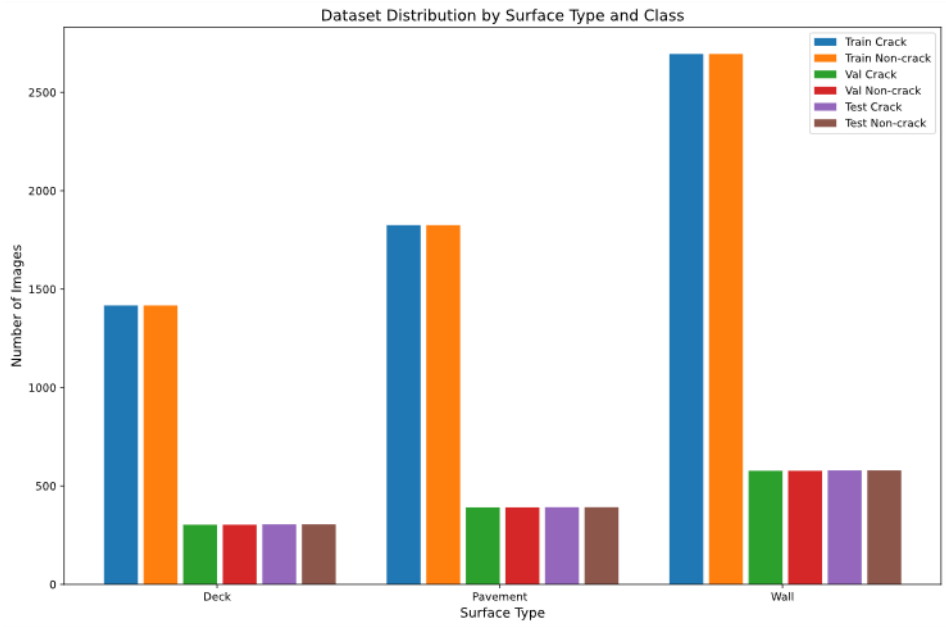

Figure 2. Graphical distribution of the dataset showing the number of images per class and data split for each surface type.

CRACKS

Importance of Cracks in Concrete Structures

Cracking is a natural, expected, and accepted characteristic of concrete; however, cracks can affect the appearance, functionality, durability, service life, or more seriously the structural integrity of concrete. For these reasons, designers, concrete producers, and contractors always strive to control or minimize the amount and severity of cracking in concrete. Nevertheless, particularly in conventionally reinforced or reinforced concrete, crack-free concrete is rarely achievable.

Fundamentally, concrete cracks. Project specifications typically require cracks in concrete to be repaired. A crack repair procedure may be pre-specified or carried out under the guidance of an engineer. Regardless of specification requirements, especially for elevated structures, cracks should be investigated before any repairs are designed or implemented. Otherwise, the repairs may fail to address the root cause of the cracking, leading to ineffective solutions that fail prematurely or do not restore the structure's original condition. More importantly, a proper investigation can determine whether a crack is an early indication of a serious issue such as a flaw in design, detailing, or construction that may compromise the structure's load-bearing capacity.

Crack formation is a fundamental characteristic of structural concrete and a central concern in the condition assessment of reinforced concrete structures [36–39]. Because concrete is a brittle material and its tensile strength is significantly lower than its compressive strength, cracking in concrete members subjected to tension is inevitable. Structural cracking can only be entirely prevented through full prestressing [40]. Reinforcement is designed and detailed to control cracking in regions where tensile stresses are expected, thereby promoting the formation of distributed and acceptably narrow cracks. Crack patterns depend on the diameter, spacing, relative rib area, and surface configuration of the reinforcing bars (whether steel or FRP), and cracks generally follow the stress trajectories

generated by the load path [41–43]. Cracking due to flexure or tension in structural members can influence both structural behavior and durability. From a serviceability perspective, cracks have the following impacts: 1) reduction of member stiffness, leading to increased deformations; 2) increased permeability, resulting in uncontrolled leakage through cracks when water is present; and 3) deterioration in the aesthetic appearance of concrete surfaces. Structural cracking is also closely associated with corrosion of embedded reinforcement, which has long been a key focus of durability research. In terms of reinforcement corrosion, permeability quantified by various diffusion coefficients is the most important performance parameter of concrete [38]. As crack width increases, both the diffusion coefficient of cracked concrete [44] and the associated flow rate [45] also increase.

Crack Width

Crack width is typically measured on the exterior surface of concrete structures. Design codes also restrict surface crack widths to mitigate serviceability and durability risks. However, crack width varies within the concrete cover, and the width along the reinforcement surface differs from that observed on the external surface of the member. Therefore, limiting crack width requires an understanding of crack geometry inside the concrete cover. Very few studies in the technical literature have examined how crack width changes through the cover thickness, and to date no reliable relationship between the crack width at the reinforcement surface and that at the concrete surface has been fully established.

In modeling structural crack widths, two fundamental assumptions appear in the literature:

1. **Bond-slip approach** proposed by Saliger in 1936 (Fig. 1a) [46,47], and
2. **No-slip approach** proposed by Broms in 1965 (Fig. 1b) [48].

Both assumptions are still used in various design codes. In 1966, Ferry-Borges [49,50] combined these concepts into a single model (Fig. 1c), later adopted in Eurocode 2 (2004 and 2023 editions) [51,52] and the fib Model Code 2010 and 2020 [53,54].

- **Bond-slip approach (Figure 3a):** Assumes a constant crack width across the concrete cover.
- **No-slip approach (Figure 3b):** Assumes zero crack width at the reinforcement level, increasing linearly toward the surface.

Neither assumption is fully acceptable, even for engineering-level simplification of structural crack widths. The combined model presumes a non-zero crack width at the reinforcement and a linear increase through the cover (Fig. 1c). While this last assumption can be used within limits for an engineering-level description of structural cracks (and is examined further in this paper), the real anatomy of cracks includes details that such simplified models cannot capture [55].

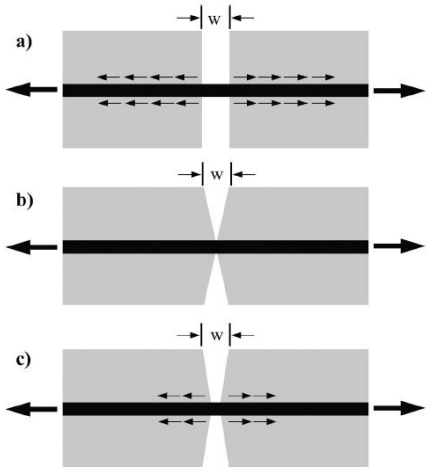

Figure 3. Schematic representation of classic crack width models, after [47], (a) The bond-slip approach, proposed in 1936 by Saliger [46], (b) The no-slip approach proposed in 1965 by Broms [48], (c). The combined model proposed in 1966 Ferry-Borges [49].

Causes of Crack Formation in Concrete

There are multiple causes of cracking in reinforced concrete structures. Although many different types of cracks may occur due to various factors, the common underlying mechanism behind most commonly observed cracks is stress. Concrete has a limited tensile capacity, both in its plastic and hardened states. It is not a ductile material; thus, it does not yield when subjected to tensile stresses. When these tensile stresses exceed the tensile strength of concrete typically around 10% of its compressive strength cracking occurs. Naturally, as freshly placed concrete hardens, its tensile strength increases. However, during the plastic and early-age stages, the tensile capacity is very low, making concrete highly susceptible to cracking during this period.

Structural Cracks

These types of cracks arise from stresses that the structure must bear due to its intended function. They usually occur in improperly designed buildings or those constructed without resolving underlying soil problems, and they are very dangerous. These cracks are not related to concrete placement or casting conditions. In such cases, authorized bodies (engineering offices, universities, etc.) should be consulted. If the structure is properly designed and not subjected to overloading, such issues typically do not arise. Structural cracks generally develop perpendicular to tensile stresses within reinforced concrete elements. For example, cracks that appear at mid-span of a simply supported beam or above the support of a cantilever beam are of this type.

Application-Induced Cracks

These cracks can occur in either fresh or hardened concrete.

Cracks in Fresh Concrete

Fresh concrete cracks typically occur between 30 minutes and 5 hours after the concrete has been placed in the formwork, most commonly in slabs or other

wide-surface applications. These cracks may reach depths of up to 10 cm, and their lengths can range from a few centimeters to as long as 2 meters. Such deep and long cracks can be extremely detrimental to the strength and durability of concrete. The two most significant causes of cracking in fresh concrete are:

- Settlement differentials, and
- Plastic shrinkage.

Settlement Cracks

These cracks typically form just above the top reinforcement in beams or in foundation concrete that has not been adequately compacted and continues to settle on its own. In fresh concrete, as the coarse aggregate particles tend to sink, water rises toward the surface. In areas directly above the reinforcement, settlement becomes more difficult, causing the concrete to move laterally toward the sides of the rebar. During this stage, if insufficient tensile strength develops, cracks form parallel to the reinforcement (Figure 4).

To prevent such cracking, it is advisable to:

- Use plastic concrete with moderate workability (not overly fluid),
- Thoroughly compact thick foundation concrete using vibrators, and
- In some cases, apply a second surface finishing (floating) about one to two hours after casting, when the surface begins to dry.
-

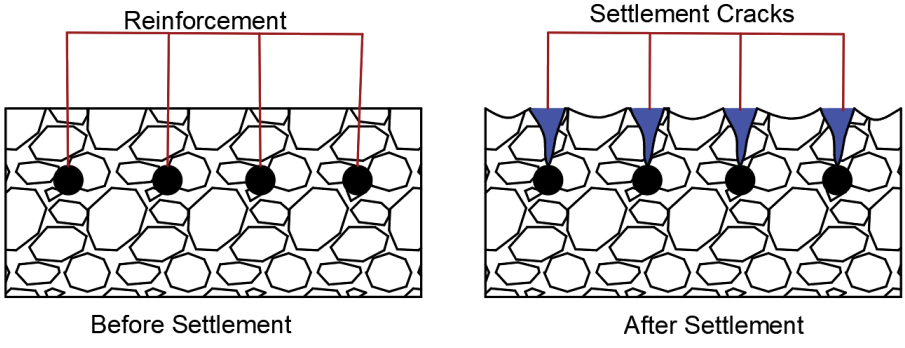

Figure 4. Settlement cracks in concrete

Plastic Shrinkage Cracks

The reduction in length and volume of concrete due to physical or chemical loss of water is referred to as shrinkage. Volume changes in concrete can be examined in three stages:

- Plastic Shrinkage
- Drying Shrinkage
- Chemical Shrinkage

How Do Plastic Shrinkage Cracks Form?

Plastic shrinkage cracks are randomly distributed surface cracks of various lengths and widths that typically appear in concrete cast under hot, dry, and windy weather conditions, especially in slabs, pavements, roadways, or airfield concrete. These cracks occur within the first few hours after concrete placement before the concrete has fully hardened and are confined to the surface layer. The term *plastic* shrinkage refers to the fact that the shrinkage occurs while the concrete is still in its plastic (i.e., moldable) state.

The main cause of plastic shrinkage is rapid evaporation of water from the concrete surface. Excess mixing water in concrete rises to the surface due to *bleeding*. If the evaporation rate exceeds the bleeding rate, the surface begins to dry and shrink. Meanwhile, the underlying, more plastic concrete cannot shrink at the same rate, resulting in tensile stresses on the surface that lead to cracking.

These cracks are usually:

- Randomly distributed
- Surface-level and shallow
- Less than 1 mm wide
- Not structurally dangerous

They are especially common in hot, dry, windy weather when casting slab-on-grade concrete or horizontal elements. Evaporation removes water from the top surface, and if the rate of evaporation is greater than the rate of bleed water rising to the surface, the surface starts drying, shrinking, and cracking.

Similar cracks may also form when freshly poured concrete is cast over a dry base, such as old concrete that hasn’t been pre-wetted, or porous materials like hollow blocks in ribbed slab systems, which absorb water from the new concrete.

CRACKS IN REINFORCED CONCRETE STRUCTURES

In seismic zones, it is essential that energy dissipation in structures is achieved in a controlled manner. In such cases, cracking is the first expected form of damage. The location, shape, width, age (new or old) of cracks, as well as deficiencies or defects in the area where they appear, provide valuable insights into the potential damage mechanisms.

Concrete is a material with high compressive strength but low tensile strength, which makes the monitoring and control of cracks in reinforced concrete members critically important [56].

Crack-related damage can be categorized as follows:

- Flexural cracks
- Shear cracks
- Torsional cracks

Flexural Cracks

Flexural cracks occur in regions where tensile stresses are highest (Figure 5). These types of cracks often indicate that the reinforcement has yielded. An example of a flexural crack in a beam is shown in Figure 6.

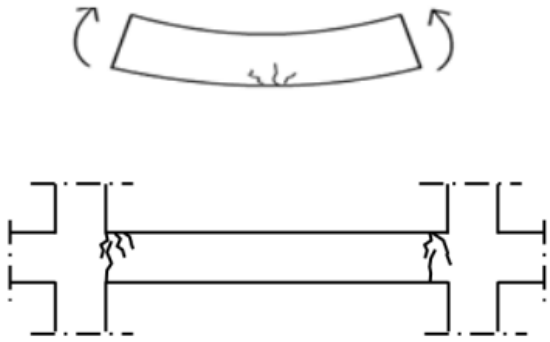

Figure 5. Flexural cracks in a reinforced concrete member [56]

Figure 6. Flexural crack in a beam [50]

Shear Cracks

Tensile cracks in beams and columns form at an angle to the beam axis (Figure 7). If the shear reinforcement is insufficient, the crack width increases. Shear cracks and the subsequent shear failure are undesirable because they represent a brittle type of failure. An example of a shear crack after an earthquake is shown in Figure 8.

If shear cracks in beams and columns are wide, it indicates severe damage. In columns, if the concrete is crushed, shear cracks are present, and the longitudinal reinforcement has buckled, the damage is considered severe [57]. Shear cracks in beams (Figure 9) and the crack formed in the reinforced concrete column-beam joint area is shown in Figure 10.

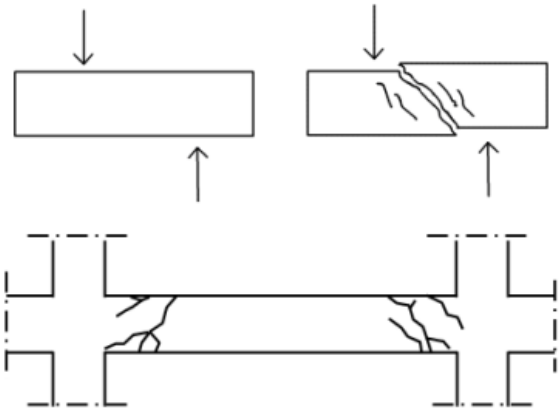

Figure 7. Shear cracks in a reinforced concrete member [58]

Figure 8. Shear cracks in beams [50]

Figure 9. Flexural cracks in reinforced concrete beams [59]

Figure 10. Cracks occurring in the beam-column joint region of reinforced concrete structures [60]

Torsional Cracks

Under torsional effects, torsional cracks form on three faces of the beam perpendicular to the principal tensile stresses (Figure 11). Additionally, crushing is observed on the fourth face. The formation of torsional cracks reduces the torsional stiffness by approximately one-tenth. As a result of this reduction, the section rotates under the nearly constant torsional moment, transferring the forces to other structural elements [57].

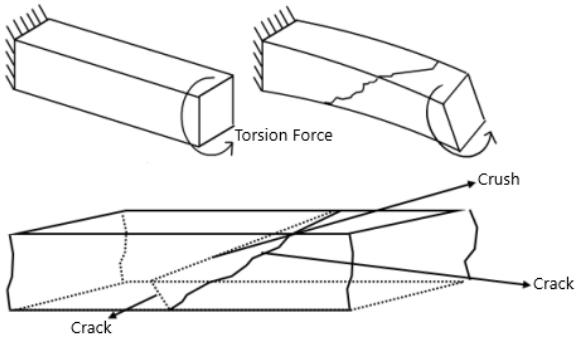

Figure 11. Torsional cracks in a reinforced concrete member [57]

Data augmentation and data preprocessing

Prior to the training of the DL models, two fundamental challenges arising from the structure of the dataset required consideration: class imbalance and the risk of overfitting stemming from limited data diversity. The raw version of the SDNET2018 dataset used in this study possessed a pronounced class imbalance, wherein the number of images belonging to the 'non-cracked' class was substantially greater than those belonging to the 'cracked' class. This situation necessitated a pre-processing step to prevent the model from developing a bias towards the majority class during training. The first and most fundamental step taken in this regard was to balance the dataset using the random down sampling method. Within this process, the number of samples in the 'cracked' class was taken as a reference, and an equal number of images were randomly selected from the more populous 'non-cracked' class, with the remainder being discarded. Through this one-time balancing operation, a fair and unbiased foundational training dataset was created, ensuring the model gives equal importance to both classes.

Although a numerical balance between the classes in the dataset was established, this state does not render the model entirely immune to the risk of overfitting. DL models possess the potential to memorize even the existing samples in a balanced but limited-diversity dataset. To eliminate this risk and to maximize the model's generalization capability, meaning its performance on previously unseen data, a second strategy of dynamic on-the-fly data augmentation was engaged. The purpose of this approach is to artificially enrich the foundational dataset created by the balancing process through various transformations applied instantaneously during training. Consequently, the model encounters a geometrically or photometrically altered version of the same image in each epoch. This continuous variation prevents the model from becoming overly reliant on the specific details of particular training examples and promotes the learning of more general and robust features.

This comprehensive data augmentation strategy incorporates both geometric and photometric transformations. During training, random geometric transformations were applied to each image in the balanced dataset. These transformations consist of

"Random Resized Crop," which encourages learning from different regions and scales of the image; "Random Horizontal Flip," applied with a 50% probability to achieve orientation invariance; and the deliberate exclusion of vertical flipping to preserve structural properties. In addition to geometric diversity, the "Color Jitter" technique was also employed, which randomly alters properties such as the brightness, contrast, and hue of the images to simulate real-world variations in lighting conditions. As a result, this two-stage approach, which first establishes a static class balance through random down sampling and then provides dynamic diversity through on-the-fly data augmentation, was designed to train robust, reliable models with high generalization capability [61,62].

Transfer Learning

The training of all DL models employed in this study leveraged the Transfer Learning approach, a cornerstone of the modern computer vision field. Transfer learning is based on the principle of reusing the knowledge and experience gained in one domain (the source domain) to solve a problem in a different but related domain (the target domain). In the context of deep learning, this typically means transferring the rich knowledge learned by a model trained on a large-scale dataset of general-purpose images, such as ImageNet, to a more specialized task, which in this study is the detection of concrete cracks. These pre-trained models have already developed a hierarchical visual understanding, extending from low-level visual features like edges, corners, and textures to more complex patterns and object parts. By using this foundational visual infrastructure as a starting point, transfer learning circumvents the challenges, high computational costs, and extensive dataset requirements associated with training a model from scratch [63,64].

The adoption of the transfer learning method in our project was a deliberate decision that provides a series of strategic advantages, rather than being merely a matter of convenience. Primarily, this approach significantly accelerates the training process and makes it more efficient. Instead of attempting to learn fundamental visual features from random weights, the models focus on learning the more complex

patterns specific to cracks on top of the general features they already know, which allows the model to converge to the targeted performance much more rapidly. Secondly, transfer learning substantially enhances model performance and generalization capability, especially on limited datasets like ours. The pre-learned features act as a powerful regularize, preventing the model from overfitting to the training data and helping it to exhibit a more consistent performance on previously unseen test data. Lastly, this method facilitates access to the most advanced (state-of-the-art) architectures with millions of parameters, such as ResNet-152, DenseNet-201, or large Transformer-based models; training such deep networks from scratch on a domain-specific dataset of limited size is practically infeasible [65–67].

In this study, transfer learning was meticulously implemented using the "Fine-Tuning" strategy, which is a widely accepted practice in the literature. This methodology begins with loading a model pre-trained on ImageNet. The architecture of the loaded model consists of two main parts: the deep convolutional layers that extract general features (the feature extractor) and the final layers that predict one of the 1000 ImageNet classes (the classifier head). Since our problem is a binary classification task ('cracked' and 'non-cracked'), the model's original 1000-class head is removed and replaced with a new binary classifier head initiated with random weights. During the training process, while the weights of this newly added head are learned freely, the weights of the pre-trained feature extractor are also "unfrozen" and updated with a much lower learning rate. This delicate process allows the model to gently adapt the general texture and edge information learned from ImageNet to the specific visual characteristics of concrete cracks [68,69].

The efficient, consistent, and reproducible execution of this complex and large-scale experimental process was made possible in large part by the use of the timm (PyTorch Image Models) library. Timm is an exceptionally comprehensive library that provides dozens of different and up-to-date computer vision architectures, such as ResNet, DenseNet, EfficientNetV2, ViT, Swin Transformer, and BEiT, along with their verified pre-trained weights. This library played a critical role in our goal of systematically training and comparing the wide array of models that form the basis

of this project. Timm offered the ability to load a desired model and easily replace its classifier head with a single line of code, and it provided a consistent framework across different architectures, thereby increasing the efficiency and scientific validity of our experiments. Consequently, this library was an indispensable tool for the scale and methodological rigor of our project.

Model Architectures

A wide array of state-of-the-art DL models, representing different architectural paradigms, was employed to solve the problem of concrete crack detection. To enhance the scope and power of the comparative analysis, architectures were selected from both CNN based approaches, which have dominated the computer vision field for over a decade, and Transformer-based architectures, which have ushered in a new era in the field. These two approaches are based on fundamentally divergent philosophies for processing visual data. CNNs, with designs inspired by the human visual cortex, focus on processing patterns in the local neighborhoods of an image in a hierarchical fashion; conversely, Transformers, adapted from their successes in natural language processing, handle an image in a holistic context, modeling the global relationships between all of its parts. This section provides a detailed explanation of the working principles, relative advantages, and differences of these two foundational architectures, thereby setting forth the strategic rationale behind the model selections in this project.

CNNs have been accepted as the de facto standard in image recognition tasks for many years. The foundation of their success lies in their specialized architectural structures that effectively utilize the spatial hierarchy and locality inherent in image data. The main building block of CNNs is the convolutional layer, which contains "convolutional filters" or "kernels." These learnable filters slide across the image to detect low-level features such as edges, corners, color gradients, and textures. As the layers deepen, these simple features learned in the preceding layers are combined to form more complex and semantically rich representations, such as an eye, a wheel, or a brick pattern. One of the greatest advantages of CNNs is their possession of

strong "inductive biases," such as "parameter sharing" and "translation invariance." Parameter sharing, which involves using the same filter across the entire image, dramatically reduces the model's parameter count and computational cost. This structure also enables the model to recognize an object regardless of its position in the image. These built-in biases allow CNNs to be highly efficient and effective, particularly on grid-like data such as images, and enable them to achieve strong generalization capabilities even with relatively less data. A simple CNN architecture is shown in Figure 12.

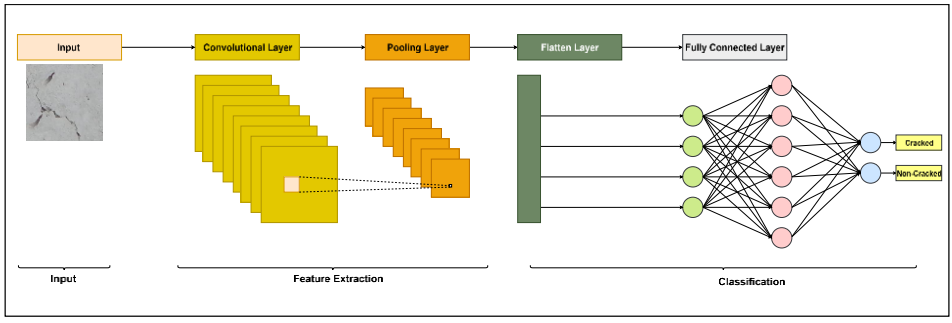

Figure 12. Basic CNN architecture

Conversely, the Transformer architecture, which was initially developed for natural language processing tasks and revolutionized that field with its superior ability to model long-range dependencies in text, has in recent years been adapted for computer vision, challenging the hegemony of CNNs. Models such as the ViT do not employ local convolutional operations in the manner of CNNs. Instead, a ViT partitions an image into a sequence of fixed-size "patches" and processes these patches as a sequence, analogous to the words in a sentence. At the heart of this architecture lies the "self-attention mechanism," which dynamically weighs the contextual importance and relationship of each patch with every other patch in the sequence. This mechanism enables Transformers to form a global contextual awareness from the very first layers and to easily model long-range relationships, such as a crack extending from one end of an image to the other, or

texture similarities between distant regions. The greatest advantage of Transformers is that they are unencumbered by the constraints of the local receptive fields found in CNNs, thereby allowing them to develop a more flexible and holistic visual understanding. This flexibility, however, means that they lack the strong built-in biases of CNNs, which generally necessitates their pre-training on much larger-scale datasets to learn effectively. Let us now examine the models utilized in our study.

ResNet

Considered a revolution in the history of CNNs, ResNet (Residual Networks) was designed to solve a fundamental problem known as "degradation," which was encountered in practice despite the theoretical expectation that model performance should increase with network depth. Beyond a certain depth, the performance of traditional CNNs would not only saturate but, on the contrary, would begin to decline rapidly. The primary reason for this issue was the difficulty that very deep networks had in learning at least an identity function with their newly added layers, thereby struggling even to preserve the performance of their shallower counterparts. The ingenious innovation from ResNet to overcome this barrier is the architectural unit called the "residual block," which contains a "shortcut connection." This shortcut connection bypasses one or more layers and directly adds their input to their output. This re-frames the learning objective: instead of learning a complex transformation function H(x) directly, the network focuses on learning a simpler residual function, $F(x) = H(x) - x$. If an added block of layers does not learn a useful feature, it becomes much easier for the network to drive the weights of that block towards zero, making F(x) zero and passing the input (x) directly to the output. This simple yet exceptionally effective mechanism made it possible for networks to reach depths of hundreds of layers, setting a new performance standard in the field of computer vision. The ResNet-34 Architecture is shown in Figure 13 [70]

.

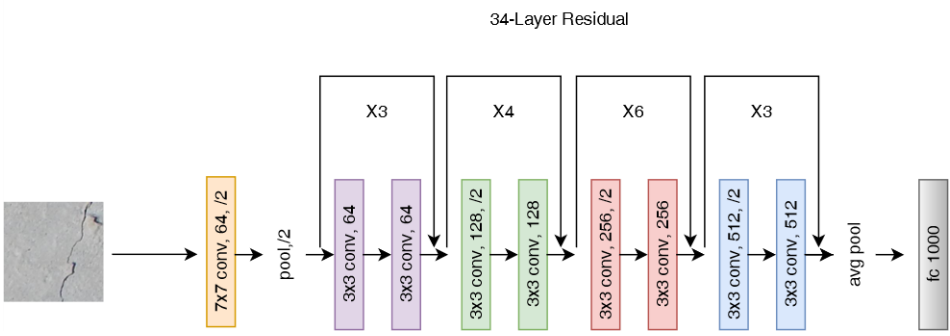

Figure 13. Architectural diagram of the ResNet-34 model, illustrating its sequential stages and the structure of its basic residual blocks.

The less deep members of the ResNet family, ResNet-18 and ResNet-34, are built upon a structure known as the "basic block," which reflects the core philosophy of the architecture in its purest form. This block consists of two sequential convolutional layers, each with a 3x3 filter size. The shortcut connection bypasses this two-layer block and merges directly with its output. The fundamental difference between ResNet-18 and ResNet-34 is their total depth, which arises from repeating these basic blocks a different number of times within the network. ResNet-18, containing fewer blocks, is a faster model that requires less computational power, whereas ResNet-34 is constructed with more blocks and is therefore a deeper network with more parameters and a higher learning capacity. The inclusion of these two models in our study allows us to analyze the relative performance and efficiency of different depth levels of the basic residual block structure on the specific task of recognizing fine-grained patterns like concrete cracks. Through this, the objective is to gain valuable insights into what level of model depth is sufficient or optimal for the complexity of the task.

As network depth increases further, the computational cost and parameter count of the basic blocks used in ResNet-34 begin to become inefficient. To solve this problem, a much more efficient block design, known as the "bottleneck," was utilized in the deeper variants of ResNet: ResNet-50, ResNet-101, and ResNet-152. This bottleneck block consists of three convolutional layers instead of two 3x3 layers: the

first 1x1 convolution creates a "bottleneck" by reducing the number of channels (depth); the middle 3x3 layer learns spatial features in this smaller dimension; and the final 1x1 convolution restores the number of channels to its original dimension. This design effectively keeps the parameter count and computational load under control while the network's depth increases significantly. The difference between ResNet-50, 101, and 152 again arises from the number of times these efficient bottleneck blocks are repeated throughout the network. ResNet-50 is the entry-level for this design, while ResNet-101 and especially ResNet-152 offer immense depth and representational power. The role of these three models in our study is to investigate the extent to which the increased capacity afforded by extreme depth and the bottleneck architecture provides a performance increase on a sensitive task like crack detection, or whether it leads to potential issues such as overfitting. The information of the layers described above is shown in Figure 14.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 14. A comparative overview of the architectures for the ResNet variants, from ResNet-18 to ResNet-152.

DenseNet

Advancing the philosophy of shortcut connections introduced by ResNet to improve information flow within the network, DenseNet (Densely Connected Convolutional Networks) presents a radical connectivity strategy to enhance network efficiency and performance. Whereas the residual blocks of ResNet add

the input to the output of the layers, the core innovation of DenseNet is its redefinition of the relationship between layers through a mechanism called "dense connectivity." Within this architecture, each layer inside a "dense block" receives the feature maps produced by *all* preceding layers as its input. Instead of being summed, these inputs are concatenated along the channel dimension. Subsequently, the layer produces its own feature map, which is then passed on to *all* subsequent layers. This structure facilitates maximum "feature reuse" within the network. Even simple features learned in the early layers, such as edges, remain directly accessible to the final layers of the network, thereby creating a collective body of knowledge. The DenseNet architecture can be seen in Figure 15 [71].

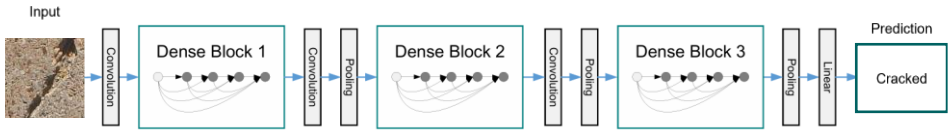

Figure 15. High-level architectural representation of a typical DenseNet, highlighting its densely connected blocks and transition layers.

The fundamental building blocks of the DenseNet architecture are the "Dense Blocks," where the dense connections occur, and the "Transition Layers," which are situated between these blocks. The dense blocks implement the aforementioned dense connectivity model, allowing for the accumulation of features across layers. A crucial hyperparameter of these blocks is the "growth rate" (k), which determines how much new information (in the form of feature maps) each layer produces. A low growth rate allows the network to be more compact and parameter-efficient. However, since the number of channels increases with each layer, this structure could lead to a rapid increase in computational cost if left unchecked. This is where transition layers become essential. These layers, placed between two dense blocks, typically use a 1x1 convolution to reduce the number of channels (acting as a bottleneck) and a 2x2 average pooling layer to down sample the spatial dimensions

(width and height) of the feature maps. This ensures that the network's general structure remains hierarchical and that it is computationally manageable.

In this study, three different variants of the DenseNet architecture with increasing depth and capacity were selected to analyze their effects on crack detection performance: DenseNet-121, DenseNet-169, and DenseNet-201. All of these models are based on the dense block and transition layer structure described above. The fundamental difference among them stems from the number of layers contained within each of the four main dense blocks, and the number in the model's name signifies the total number of learnable layers in the network. For instance, while DenseNet-121 has blocks containing fewer layers, the dense blocks of DenseNet-201 contain many more layers, which grants it a higher learning capacity and representational power. The analysis of these three variants aims to reveal how effective the dense connectivity and feature reuse strategy, applied at different depth levels, is in the task of recognizing fine and complex textural patterns like concrete cracks, and what advantages this might offer compared to the additive approach of ResNet. You can see the aforementioned layers in Figure 16.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figure 16. Comparative architectural specifications of the DenseNet variants (121, 169, 201, and 264).

Efficientnet-V2

Traditionally, enhancing the performance of CNNs was typically achieved by focusing on a single dimension, such as increasing the network's depth (more layers), width (more channels), or the resolution of the input image. These approaches, however, often yield diminishing returns and disregard the need to establish a delicate balance among these three dimensions to achieve optimal performance. EfficientNet introduced a revolution by providing a systematic solution to this problem with a method it termed "compound scaling." The core philosophy of this method is to scale the network's depth, width, and resolution not arbitrarily and separately, but simultaneously and in a balanced manner through a fixed mathematical relationship. This allows the model to grow in the most efficient way possible, both in terms of parameter count and computational cost (FLOPs). By using this principle, EfficientNet set a new standard for efficiency and performance, achieving similar or higher accuracy than other state-of-the-art models of its time with significantly fewer parameters [72].

Although the first version of EfficientNet was groundbreaking in terms of parameter efficiency and accuracy, it presented some practical challenges, such as long training times, especially for larger models. EfficientNetV2 is a next-generation architecture developed to address these training bottlenecks and to further advance both training speed and efficiency. One of the primary innovations of EfficientNetV2 is the replacement of the standard MBConv blocks in the early stages of the architecture with "Fused-MBConv" blocks, which operate more efficiently on modern hardware (GPUs/TPUs), thereby increasing training speed. Its second and most important innovation is a dynamic training strategy called "progressive learning." In this strategy, the model is initially trained with smaller image sizes and weaker data augmentation (regularization) techniques. As training progresses, both the size of the input image and the intensity of the augmentation are gradually increased. This approach allows the model to learn simple patterns quickly at the start while enabling it to focus on

more complex features in the later stages, consequently shortening the total training time considerably.

In this study, three primary variants of the EfficientNetV2 family were utilized to analyze the effect of the efficiency and performance balance it offers at different scales: EfficientNetV2-Small (S), EfficientNetV2-Medium (M), and EfficientNetV2-Large (L). These models share the same fundamental architectural principles and are scaled according to different values of the compound scaling coefficient. EfficientNetV2-S is a highly efficient model that requires fewer parameters and computational resources, generally presenting an ideal balance for resource-constrained situations. EfficientNetV2-M offers a significant performance increase over the "Small" version, but it does so with a reasonable increase in parameters. EfficientNetV2-L, on the other hand, is the member of the family that targets the highest performance and expands the architecture on a large scale to achieve state-of-the-art accuracy rates. The analysis of these three variants allows for an evaluation of whether the computational cost of achieving the highest accuracy is necessary for a specific task like concrete crack detection, or if a more efficient model offers a sufficient and more suitable solution for practical applications.

Vision Transformers

The ViT, which represents a fundamental paradigm shift among the architectures examined in this study, is predicated on a philosophy radically different from the previously discussed convolution-based approaches. ViT adapts the powerful Transformer architecture, whose origins lie in revolutionizing the field of Natural Language Processing (NLP) for processing text data, to computer vision tasks. In contrast to CNNs, which view an image as a structure to be processed hierarchically through local neighborhoods of pixels, a ViT treats an image as a sequence of "patches," analogous to how a sentence is composed of words. This approach discards the inductive biases inherent in CNNs, such as locality and translation invariance, which can constrain the

model's learning process. Instead, ViT attempts to learn the relationships between all parts of an image from scratch, directly from the data. This allows it to offer exceptional flexibility and potential for modeling complex and long-range contextual relationships, even between distant points in an image.

The operation of the ViT architecture consists of steps that convert an image into a series of vectors and subsequently process this sequence with a Transformer encoder. In the first step, the 224x224 input image is divided into a non-overlapping, fixed-size grid; each cell of this grid is a "patch." For example, in a patch_16 model, the image is partitioned into 196 patches of 16x16 pixels. Each patch is then flattened into a one-dimensional vector and passed through a learnable linear projection layer to create "patch embeddings" suitable for the model's working dimension. Since the Transformer architecture has no inherent sense of sequence, "positional embeddings" are added to these vectors to preserve the original location information of the patches. The resulting sequence of vectors is then fed into a standard Transformer encoder, which is the heart of the architecture. This encoder contains "Multi-Head Self-Attention" layers that dynamically weigh the importance and relationship of each patch with every other patch. This mechanism allows the model to learn a holistic representation of the image from the very first layers. The Vision Transformers architecture is shown in Figure 17 [73].

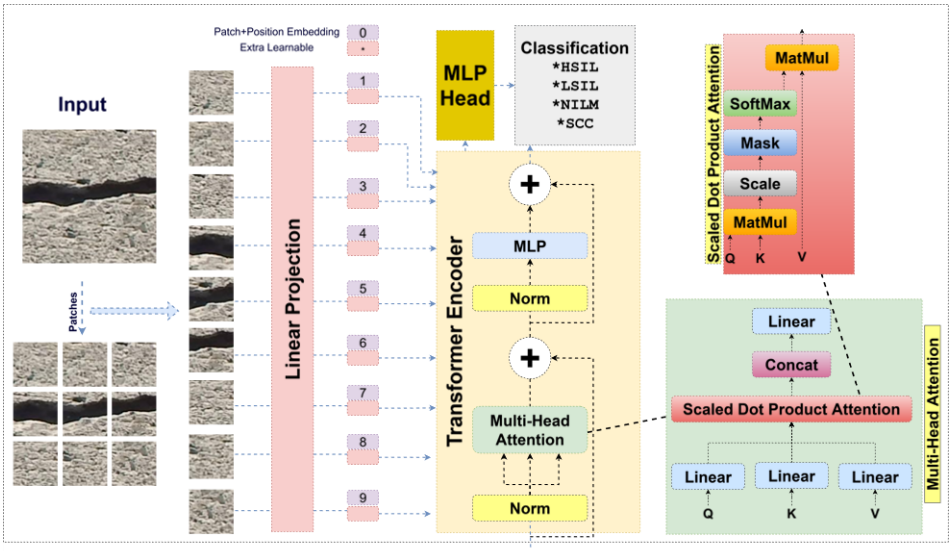

Figure 17. Detailed architectural diagram of the ViT, illustrating the key components from image patching and embedding to the inner workings of the Transformer Encoder and the final classification head.

The ViT models utilized in this study are differentiated according to two fundamental hyperparameters: patch size and model scale. The patch size is a critical factor that dictates the resolution at which the model processes an image and determines the length of the input sequence. In this project, two different patch sizes were tested: patch_16 and patch_32. The patch_16 models partition the image into smaller pieces, thereby creating a longer vector sequence. This enhances the model's potential to capture finer and more detailed features, such as narrow cracks; however, since the computational cost of the self-attention mechanism is proportional to the square of the sequence length, it slows down the training process and requires more memory. On the other hand, patch_32 models use larger patches, which creates a shorter sequence, making them much more efficient and faster in terms of computation. This efficiency, however, comes with the risk of processing the image at a coarser resolution and potentially losing small details. Testing these two different patch sizes is of critical

importance for understanding the trade-off between detail sensitivity and computational efficiency for the crack detection task.

In addition to patch size, ViT architectures can also be scaled to possess different levels of capacity and complexity. In this study, various industry-standard model variants, designated as Tiny, Small, Base, and Large, were used for both patch sizes. The fundamental differences among these variants stem from the number of Transformer encoder layers (depth), the size of the patch embeddings (hidden size/width), and the number of heads in the self-attention mechanism. The Tiny and Small models offer lighter and more efficient alternatives with fewer layers and a lower parameter count, while the Base model is the standard configuration defined in the original ViT paper, providing a strong performance baseline. The Large model, conversely, is the highest-capacity member of the family with significantly more layers and parameters, and it targets state-of-the-art accuracy at the expense of the highest computational cost. This wide range of models allows us to comprehensively investigate the capabilities of the ViT architecture across different configurations and to determine the most suitable model complexity for the task.

Swin

Although the standard ViT architecture is successful at modeling long-range dependencies by virtue of its global self-attention mechanism, it presents a significant practical challenge: its computational cost increases quadratically with the number of input image patches (quadratic complexity). This characteristic renders its use highly expensive for tasks that require high-resolution images, such as object detection or semantic segmentation. The Swin (Shifted Window) Transformer was developed precisely to solve this efficiency problem and to establish the Transformer architecture as a more general-purpose computer vision backbone. The core innovation of the Swin Transformer is its replacement of ViT's costly global attention mechanism with a local attention mechanism that is much more computationally efficient and operates within a

hierarchical structure. In this approach, self-attention is computed not among all patches in the image, but within smaller, non-overlapping local windows. As a result, the computational complexity exhibits a linear, rather than quadratic, increase with respect to the image size, which allows the architecture to operate efficiently with high-resolution inputs [74]. The Swin Transformers architecture is shown in Figure 18.

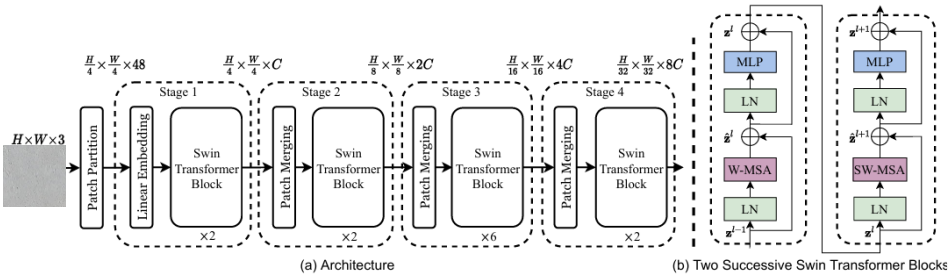

Figure 18. An overview of the Swin Transformer architecture, illustrating (a) the general hierarchical structure with four stages and patch merging layers, and (b) the detailed composition of two successive blocks featuring windowed (W-MSA) and shifted-window (SW-MSA) self-attention.

The most ingenious aspect of the Swin Transformer is its ability to increase efficiency using local windows without losing global context, which it achieves by enabling information flow across windows. This is realized through the "shifted window" mechanism. In the first of two successive Transformer blocks in the architecture, self-attention is computed within standard, non-overlapping windows. In the next block, however, this window grid is shifted by half the window size. This simple shifting operation allows patches that were in different windows in the preceding layer to be grouped together within the same window in the new layer. As this process is repeated throughout the depth of the network, information effectively propagates from one window to another, and consequently, a global interaction field is achieved through local operations. Another significant advantage of this design is its ability to produce hierarchical

feature maps at different scales, similar to CNNs. At different stages of the network, it reduces the spatial resolution and increases the channel count by merging patches. This hierarchical structure makes the Swin Transformer an extremely powerful and flexible backbone not only for classification but also for various dense prediction tasks.

In this study, four primary variants of the Swin Transformer family were used to evaluate the effect of its efficiency and performance balance at different capacity levels: Swin-Tiny (T), Swin-Small (S), Swin-Base (B), and Swin-Large (L). All of these variants are based on the shifted-window self-attention mechanism and hierarchical design described above. The fundamental difference among them arises from the scaling of the core parameters that constitute the architecture; these parameters include the number of blocks at each stage (depth), the size of the embedding vectors (channel count), and the number of heads in the attention mechanism. Swin-T and Swin-S, as the lighter and more efficient members of the family, offer the fundamental advantages of the Swin architecture at a lower computational cost. Swin-B is the reference model, which establishes a strong balance between performance and cost and is often used for direct comparison with standard-sized models like ViT-Base. Swin-L, in contrast, is the largest model in terms of capacity, targeting the highest accuracy. The analysis of these four variants allows us to understand how effective a hierarchical and efficient Transformer approach is for a specific task like concrete crack detection and the impact of model capacity on performance. The layers and architectural variants mentioned above are shown in Figure 19.

	downsp. rate (output size)	Swin-T	Swin-S	Swin-B	Swin-L
stage 1	4× (56×56)	concat 4×4, 96-d, LN	concat 4×4, 96-d, LN	concat 4×4, 128-d, LN	concat 4×4, 192-d, LN
		[win. sz. 7×7, dim 96, head 3] × 2	[win. sz. 7×7, dim 96, head 3] × 2	[win. sz. 7×7, dim 128, head 4] × 2	[win. sz. 7×7, dim 192, head 6] × 2
stage 2	8× (28×28)	concat 2×2, 192-d , LN	concat 2×2, 192-d , LN	concat 2×2, 256-d , LN	concat 2×2, 384-d , LN
		[win. sz. 7×7, dim 192, head 6] × 2	[win. sz. 7×7, dim 192, head 6] × 2	[win. sz. 7×7, dim 256, head 8] × 2	[win. sz. 7×7, dim 384, head 12] × 2
stage 3	16× (14×14)	concat 2×2, 384-d , LN	concat 2×2, 384-d , LN	concat 2×2, 512-d , LN	concat 2×2, 768-d , LN
		[win. sz. 7×7, dim 384, head 12] × 6	[win. sz. 7×7, dim 384, head 12] × 18	[win. sz. 7×7, dim 512, head 16] × 18	[win. sz. 7×7, dim 768, head 24] × 18
stage 4	32× (7×7)	concat 2×2, 768-d , LN	concat 2×2, 768-d , LN	concat 2×2, 1024-d , LN	concat 2×2, 1536-d , LN
		[win. sz. 7×7, dim 768, head 24] × 2	[win. sz. 7×7, dim 768, head 24] × 2	[win. sz. 7×7, dim 1024, head 32] × 2	[win. sz. 7×7, dim 1536, head 48] × 2

Figure 19. Detailed specifications of the Swin Transformer variants (Swin-T, Swin-S, Swin-B, and Swin-L), showing the hyperparameter configuration at each hierarchical stage.

Beit

Among the Transformer-based models examined in this study is BEiT (Bidirectional Encoder representation from Image Transformers), which presents not an architectural innovation, but a revolutionary pre-training strategy that fundamentally alters how models learn visual representations. Traditionally, models like ViT and Swin are pre-trained in a supervised fashion on datasets such as ImageNet, which contain millions of human-labeled images. BEiT, conversely, adopts a self-supervised learning approach that circumvents the need for this costly and labor-intensive labeling process. Drawing inspiration from the groundbreaking BERT model in the field of Natural Language Processing (NLP), the core idea behind BEiT is for the model to learn visual representations autonomously through a "pretext task" known as "Masked Image Modeling" (MIM). In this process, an input image is first converted into a sequence of patches and, subsequently, into a version composed of discrete "visual tokens." During pre-training, a portion of the image patches (e.g., 40%) is randomly masked, and the model's task is to predict the original visual tokens of these masked regions by using the context of the visible patches. This task compels the model to do much more than simple pixel completion; it forces a deep

understanding of the semantic and structural integrity of images [75]. The Beit Transformers architecture is shown in Figure 20.

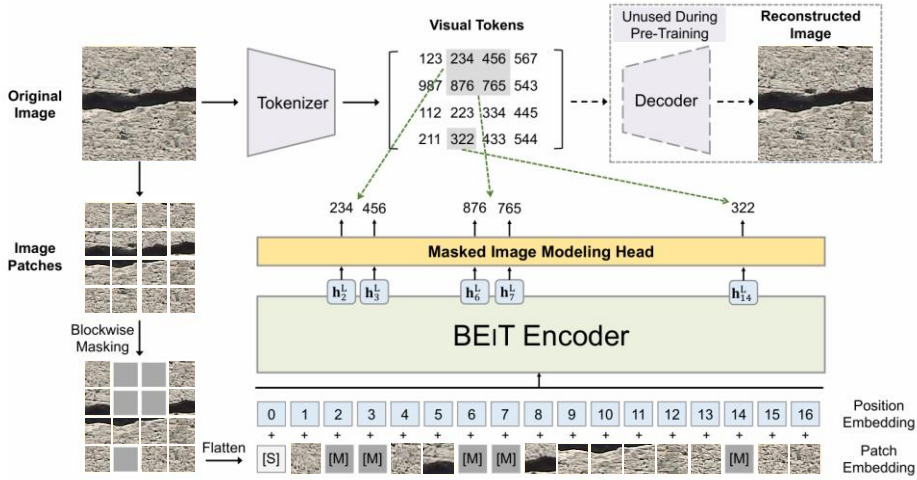

Figure 20. A schematic diagram of the BEiT pre-training framework, illustrating its core task of Masked Image Modeling (MIM).

Since BEiT itself is a pre-training methodology rather than a novel architecture, the backbone it is based upon is typically a standard Vision Transformer. The BEiT-Base and BEiT-Large variants used in this study denote different capacity versions of the underlying ViT architecture, upon which the masked image modeling task was applied. BEiT-Base is constructed upon a standard "Base" size ViT architecture (e.g., 12 layers, 768 hidden dimension, 12 attention heads). This model offers a strong baseline for measuring the efficacy of the self-supervised pre-training on a standard-sized architecture. BEiT-Large, conversely, uses a "Large" size ViT architecture with significantly more layers and parameters, aiming to maximize the potential of this learning strategy and generally targeting state-of-the-art performance. The primary purpose of including these two models in our study is to compare not only architectures but also pre-training strategies. The comparison of BEiT models against traditionally supervised, ImageNet pre-trained ViT models of the same size is intended to

determine whether learning from unlabeled data with a BERT-like approach provides a tangible advantage over supervised pre-training for a specific task like concrete crack detection, where fine details are important.

RESULTS AND DISCUSSION

Experimental Setup

To ensure that the findings presented in this research adhere strictly to the principles of scientific rigor, methodological consistency, and reproducibility, all experimental work was conducted on a single, meticulously configured, and controlled computing platform. This standardized approach eliminates external variables, such as performance fluctuations arising from hardware or software, which could otherwise confound the results. Thus, it is guaranteed that any observed differences in performance, such as accuracy, training time, and efficiency, can be reliably attributed solely to the intrinsic architectural merits and structural properties of the DL models under investigation. This methodological rigor is an indispensable prerequisite for the primary objective of the study: a fair comparison of different architectural paradigms on a level playing field.

The experimental work was conducted on a high-performance workstation built upon an ASUS PRIME Z790-A WIFI motherboard. At the heart of the system is an NVIDIA GeForce RTX™ 3090 graphics processing unit (GPU), which, with its 24 GB of GDDR6X VRAM, undertakes the majority of the DL computations and makes the training of large models possible. This extensive VRAM capacity, required by high-parameter model variants such as the "Large" versions in our study and by large batch sizes, ensured that the model training could proceed uninterruptedly and efficiently. Tasks such as data loading, pre-processing, on-the-fly data augmentation, and general system management were handled by a powerful Intel® Core™ i7-14700K central processing unit (CPU), which ensures the GPU operates at full efficiency without data bottlenecks. These two processors are supported by 64 GB of DDR5 system memory and a 1.0 TB high-speed storage unit, which facilitate the fluid operation of the system and the efficient management of large data batches in memory.

On top of the hardware layer resides the Ubuntu 24.04.2 LTS operating system (Linux 6.11.0-28-generic kernel version), which was chosen for its stability and strong compatibility with DL tools. All model development, training, and evaluation processes were coded in Python using the PyTorch DL framework, a popular choice among researchers due to its flexible structure, dynamic computational graph, and rich ecosystem. PyTorch's ability to fully leverage the parallel computing power of the RTX 3090 was enabled by the CUDA 12.8 toolkit, which serves as the essential bridge between the hardware and software. In effect, this integrated experimental environment, meticulously managed and standardized from the operating system down to the lowest-level hardware drivers, provides the robust, stable, and verifiable foundation for the comprehensive comparative analysis conducted in this research.

Evaluation Metrics

To quantitatively and comprehensively evaluate the performance of the DL models trained in this study, a series of industry-standard metrics were employed. All of these metrics are computed based on the Confusion Matrix, which is derived from a comparison of the model's predictions on the test set against the ground-truth labels. The confusion matrix is a table that visualizes the performance of a classification model and is built upon four fundamental outcomes. In the context of our project, these outcomes are defined as follows: a True Positive (TP) is a 'cracked' image correctly classified by the model as 'cracked' ; a False Positive (FP) is a 'non-cracked' image erroneously labeled by the model as 'cracked' ; a True Negative (TN) is a 'non-cracked' image correctly classified as 'non-cracked' ; and finally, a False Negative (FN) is a 'cracked' image that was missed by the model and erroneously classified as 'non-cracked'. These four foundational outputs form the basis for the more complex and interpretable metrics described below.

Accuracy is the most intuitive and fundamental metric used to measure the performance of a classification model. Mathematically, it is calculated by

dividing the sum of all correct predictions (both true positives and true negatives) by the total number of predictions. In other words, it represents the percentage of images in the test set that the model classified correctly. While a high accuracy value indicates that the model is performing well in a general sense, it can be misleading, particularly on imbalanced datasets. Although the dataset in this study was balanced, evaluating the model's performance not just by its accuracy rate but also with more sensitive metrics that separately examine its success on the positive and negative classes is essential for a deeper and more reliable analysis.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

The Precision metric, also known as Positive Predictive Value, quantifies what proportion of the predictions labeled as "positive" (i.e., 'cracked') by the model are genuinely correct. This metric provides an answer to the question, "When the model reports that it has found a crack, how much can I trust that finding?" A high precision score indicates that the model has a low False Positive (FP) rate. In the context of structural health monitoring, this is of critical importance because high precision prevents a sound concrete surface from being erroneously flagged as 'cracked'. This, in turn, precludes unnecessary and costly on-site inspections, detailed analyses, or repair procedures, promoting the efficient allocation of resources.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

This metric, alternatively referred to as Sensitivity, Recall, or True Positive Rate, measures the model's ability to correctly identify all actual positive cases (i.e., all images that are genuinely 'cracked'). This metric answers the question, "What percentage of the existing cracks was our model able to capture?"

Sensitivity is one of the most important metrics, particularly in safety-critical applications. A high sensitivity value signifies that the model has a low False Negative (FN) rate. In other words, the likelihood of the model overlooking an existing crack is low. Since the failure to detect an existing crack in a reinforced concrete structure can allow structural damage to grow over time, potentially leading to catastrophic consequences, maintaining a high value for this metric is of paramount importance for proactive maintenance and safety management.

$$Recall(Sensitivity) = \frac{TP}{TP + FN} \tag{3}$$

In practice, a trade-off generally exists between the Precision and Sensitivity metrics; a modification made to increase one may cause a decrease in the other. For instance, a model that labels even the slightest suspicion as 'cracked' might achieve very high sensitivity, but this situation would lead to numerous false positives, thereby reducing its precision. The F1-Score is a powerful metric that establishes a balance by combining these two metrics into a single number. Calculated as the harmonic mean of Precision and Sensitivity, the F1-Score requires both metrics to be high. The harmonic mean, unlike a simple arithmetic average, severely penalizes the F1-Score if one of the metrics is very low. For this reason, the F1-Score is regarded as one of the most balanced and reliable indicators for measuring a model's performance in situations where both false positives (low precision) and false negatives (low sensitivity) are significant.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

Training Protocol

To ensure a fair and unbiased measurement of the true performance of all state-of-the-art architectures compared in this study, all experiments were conducted under strictly standardized and identical conditions. This rigorous

standardization forms the foundation of our methodology, guaranteeing that any observed performance differences are attributable to the architectural merits of the models themselves, rather than to variations in the training setup. All models were trained and evaluated on the same hardware, using the same balanced dataset splits (training, validation, test), the same input image size (224×224), and the same batch size. This consistency was further reinforced by initializing each architecture with its own standard pre-trained weights and subjecting it to the exact same training pipeline.

Our comprehensive training strategy, applied consistently to all models, integrated the principles of transfer learning and data augmentation. The entire process was foundationally built on transfer learning. By fine-tuning pre-trained models, we leveraged powerful feature extractors that had already learned a rich hierarchy of general visual features. The issue of class imbalance was fundamentally resolved *prior* to training by balancing the dataset via down sampling. Data augmentation techniques were then used on this balanced dataset to enhance the model's generalization capability and to prevent overfitting. On-the-fly data augmentation techniques such as random rotation, flipping, cropping, and color shifting artificially increased the diversity and effective volume of the training data, making the models more robust and resilient to variable conditions.

To update the model weights, the AdamW optimizer was chosen, which is known for its stability and effectiveness in training DL architectures, especially those that are Transformer-based. AdamW tends to provide better generalization performance than the traditional Adam optimizer because it decouples the weight decay regularization from the gradient update. A dynamic approach was adopted for managing the learning rate. The training process included a "warm-up" period for the first 5 epochs, during which the learning rate was gradually increased from zero to its target value. This warm-up phase prevents the valuable pre-trained weights from being corrupted by large, abrupt updates, which can occur when the model is unstable at the beginning of training. Following the warm-up period, a Cosine Annealing Scheduler was engaged, which smoothly decays the learning

rate according to a cosine curve over the remaining epochs. The initial learning rate was set to 1e-5.

Additional mechanisms were used for managing the training duration and controlling for overfitting. In this context, the training of all models was conducted for a fixed and standard duration of 100 epochs to ensure a fair comparison. Throughout the training process, the model's performance on the validation dataset was carefully monitored at the end of each epoch using the F1-Score metric. Once the 100-epoch training was complete, the model weights belonging to the epoch that achieved the highest validation F1-Score were saved as the final, best-performing version of the respective architecture. Another important part of the regularization strategy was the application of a weight decay of 2.0e−05 as a parameter of the AdamW optimizer. This technique penalizes the magnitude of the model's parameter weights, thereby discouraging overly complex and overfitted solutions.

The training protocol applied in this study is, therefore, a multi-faceted and meticulously controlled set of strategies. The establishment of a standardized experimental environment, the creation of a strong foundation with transfer learning, the reinforcement of generalization with data augmentation, and the use of an advanced learning rate schedule with warm-up and cosine decay, along with regularization techniques like weight decay, were all aimed at revealing the potential of each architecture under fair and optimal conditions. This holistic approach ensures the reliability of the obtained results and the scientific validity of the comparative analysis.

Experimental Results

This section presents the experimental results that reveal the performance of the wide range of models examined in this study on the meticulously prepared datasets. To deepen the comparative analysis and to measure the capabilities of the models at different levels of difficulty, the experiments were conducted in two primary phases. In the first phase, each structural category—'Deck',

'Pavement', and 'Wall'—was treated as an independent binary classification (cracked/non-cracked) problem. This approach allowed us to evaluate in an isolated manner the fundamental success of each architecture in learning the textural and structural features specific to different surface types. In the second phase, to test the performance of the models in a more complex scenario, these three main categories were combined to create a single, unified dataset. This combination resulted in the definition of a new six-class multi-class classification task, comprising the following classes: D_Cracked (Cracked Deck), D_Non-cracked (Non-cracked Deck), P_Cracked (Cracked Pavement), P_Non-cracked (Non-cracked Pavement), W_Cracked (Cracked Wall), and W_Non-cracked (Non-cracked Wall). In the subsections that follow, the results obtained from this two-phase experiment are presented in detail. First, the results of the binary classification tasks are addressed, within which the performance metrics obtained for the 'Deck' class are summarized in Table 2.

Table 2. Performance comparison of all evaluated architectures on the 'Deck' dataset for the binary classification task.

Models		Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Params	GFLOPs
ResNet	18	83.61	83.75	83.61	83.59	11.18	3.6470
	34	83.61	83.98	83.61	83.56	21.29	7.3565
	50	81.48	81.80	81.48	81.43	23.51	8.2634
	101	83.11	84.31	83.11	82.97	42.5	15.7288
	152	83.93	84.72	83.93	83.84	58.15	23.2038
DenseNet	121	82.30	82.36	82.30	82.29	6.96	5.6667
	169	79.02	80.73	79.02	78.72	12.49	6.7169
	201	78.85	79.05	78.85	78.82	18.1	8.5795

EfficientNet-V2	Small		83.77	84.27	83.77	83.71	20.18	5.4192
	Medium		82.13	83.29	82.13	81.97	52.86	10.4436
	Large		84.59	85.47	84.59	84.49	117.24	23.9733
Vision Transformers	Patch-16	Tiny	82.30	82.86	82.30	82.22	5.52	2.1493
		Small	80.49	80.77	80.49	80.45	21.67	8.4817
		Base	79.34	79.39	79.34	79.34	85.8	33.6955
		Large	80.66	81.73	80.66	80.49	303.3	119.2923
	Patch-32	Small	80.00	80.70	80.00	79.89	22.49	2.2390
		Base	81.80	82.09	81.80	81.76	87.46	8.7247
		Large	80.66	81.37	80.66	80.55	305.51	30.5073
Swin Transformers	Tiny		82.95	84.19	82.95	82.80	27.52	8.7422
	Small		83.44	83.75	83.44	83.41	48.84	17.0885
	Base		84.43	84.49	84.43	84.42	86.75	30.3375
	Large		84.59	84.64	84.59	84.58	195.0	68.1649
Beit Transformers	Base		82.30	82.65	82.30	82.25	85.76	25.3294
	Large		79.84	79.87	79.84	79.83	303.41	89.5463

The results of the binary classification task for the "Deck" class revealed significant performance differences and important architectural trends among the examined model families. According to the metrics presented in Table 2, the Swin

Transformer and EfficientNet-V2 families generally exhibited the highest performance, while the classic ResNet architectures also yielded highly competitive and stable results. The highest F1-Score (84.58%) was achieved by the Swin Transformer Large model, followed very closely by the EfficientNet-V2 Large model with a score of 84.49%. These two models stand out as the most successful approaches for the classification of "Deck" class images, which are characterized by homogeneous surface textures and sharp shadows. The confusion matrices for the models that yielded the top performance for the 'Deck' class are illustrated in Figure 21.

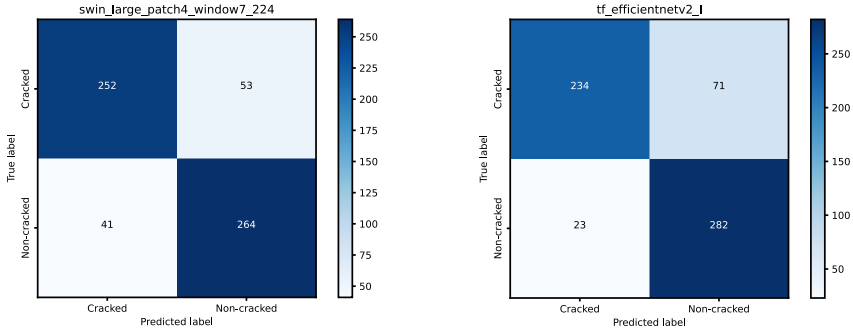

Figure 21. Confusion matrices for the two best-performing models on the 'Deck' binary classification task: Swin Transformer Large (left) and EfficientNet-V2 Large (right).

When examining the CNN-based architectures, the ResNet family is seen to offer reliable and stable performance, with F1-scores generally above 83%. The best result within the family was achieved by ResNet-152 (83.84% F1); however, a noteworthy point is that ResNet-50 exhibited a lower performance than the smaller ResNet-34. This situation is an indicator that model depth does not always bring a linear performance increase. In contrast, the DenseNet family underperformed compared to other architectures in this task. The best DenseNet model was the smallest in the family, DenseNet-121 (82.29% F1), while performance decreased as model depth increased (DenseNet-169 and 201),

showing that the dense connectivity strategy did not scale effectively for this specific task. Among the CNNs, the most striking success was demonstrated by the EfficientNet-V2 family. While EfficientNet-V2 Large achieved one of the highest scores, the EfficientNet-V2 Small model proved what a powerful alternative it is in terms of efficiency by reaching a very high F1-score of 83.71% with an extremely low computational cost (5.4 GFLOPs).

When considering the Transformer-based architectures, the differences between the architectural designs become clearly apparent. The standard ViT models obtained more modest results in general. Particularly in the ViT-Patch-16 family, the fact that the largest model, ViT-Large (80.49% F1), performed lower than the smallest model, ViT-Tiny (82.22% F1), demonstrates that simply increasing the parameter count is not an effective strategy for this task. In contrast, the Swin Transformer family, which uses a hierarchical structure and a windowed attention mechanism, was by far the most successful among the Transformers. In this family, where performance consistently increased with model size, Swin-Base (84.42% F1) and Swin-Large (84.58% F1) were among the top-performing models. Finally, although the BEiT models had a reasonable start (BEiT-Base 82.25% F1), the performance drop in the BEiT-Large model indicates that self-supervised pre-training did not provide the expected advantage for large models in this task.

Evaluating the results not only based on absolute performance but also along the axis of efficiency (parameter count and GFLOPs) offers important takeaways for practical applications. Although Swin-Large and EfficientNet-V2 Large stand out as the best options in scenarios where maximum accuracy is targeted, the efficiency champions are different. Specifically, EfficientNet-V2 Small and ResNet-34 achieve F1-scores only marginally lower than the top-tier models, but they do so at one-fifth or less of the computational cost. This situation clearly demonstrates that in real-world applications where resources are limited or fast inference time is important, lighter and more efficient models represent a much more practical and sensible alternative.

Table 3. Performance comparison of all evaluated architectures on the 'Pavement' dataset for the binary classification task.

Models			Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Params	GFLOPs
ResNet	18		90.43	90.47	90.43	90.43	11.18	3.6470
	34		90.94	90.96	90.94	90.94	21.29	7.3565
	50		90.82	90.82	90.82	90.82	23.51	8.2634
	101		90.43	90.55	90.43	90.43	42.5	15.7288
	152		90.05	90.09	90.05	90.05	58.15	23.2038
DenseNet	121		89.92	89.93	89.92	89.92	6.96	5.6667
	169		86.48	86.83	86.48	86.45	12.49	6.7169
	201		85.08	85.13	85.08	85.07	18.1	8.5795
EfficientNet-V2	Small		89.80	89.82	89.80	89.79	20.18	5.4192
	Medium		89.92	90.09	89.92	89.91	52.86	10.4436
	Large		90.05	90.06	90.05	90.05	117.24	23.9733
Vision Transformers	Patch-16	Tiny	88.27	88.27	88.27	88.26	5.52	2.1493
		Small	89.67	90.15	89.67	89.64	21.67	8.4817
		Base	88.14	88.16	88.14	88.14	85.8	33.6955
		Large	88.65	89.12	88.65	88.61	303.3	119.2923
	Patch-32	Small	87.12	87.66	87.12	87.07	22.49	2.2390
		Base	87.37	87.43	87.37	87.37	87.46	8.7247
		Large	87.24	87.53	87.24	87.22	305.51	30.5073

Swin Transformers	Tiny	87.12	87.17	87.12	87.11	27.52	8.7422
	Small	88.27	88.46	88.27	88.25	48.84	17.0885
	Base	90.43	90.51	90.43	90.43	86.75	30.3375
	Large	89.16	89.44	89.16	89.14	195.0	68.1649
Beit Transformers	Base	89.41	89.49	89.41	89.41	85.76	25.3294
	Large	75.38	76.57	75.38	75.10	303.41	89.5463

The results of the binary classification task conducted on the 'Pavement' dataset, presented in detail in Table 3, reveal a performance distribution that is different from the 'Deck' class and quite remarkable. The most successful results in this category were surprisingly achieved by the classic ResNet family, which surpassed many more modern and structurally complex architectures. It is noteworthy that the general performance scores are higher compared to the 'Deck' class and are clustered more closely around a 90% F1-Score. This situation indicates that the crack features in the pavement dataset, despite the high visual noise it contains, could be learned more clearly and distinctively by the models. The absolute winner of this category was ResNet-34, with an impressive F1-Score of 90.94%. This result is of considerable importance as it underscores the fact that the newest or largest model is not always the best solution for a given task, and that a well-established, medium-depth architecture can also deliver top-tier performance.

When the performance within the CNN-based architectures is examined, the clear superiority of the ResNet family in this category is plainly visible. The top-performing ResNet-34 was followed very closely by other members of the family such as ResNet-50, ResNet-18, and ResNet-101, which also achieved F1-scores

above 90%. This suggests that medium-depth ResNet architectures create a "sweet spot" of performance for the detection of pavement cracks, and that extreme depth, as in ResNet-152, results in a marginal performance decrease compared to this peak. In contrast, the DenseNet family, showing a similar trend to the 'Deck' results, could not demonstrate top performance in this category either. The fact that the best result was again obtained by the smallest model, DenseNet-121, and that performance declined in deeper models, has strengthened the evidence that the dense connectivity strategy does not provide a scalable advantage for the tasks in this project. The EfficientNet-V2 family, however, exhibited highly consistent and successful results, with all variants achieving very similar F1-scores (in the range of 89.8% - 90.05%). The most important finding here is that the lightest model, EfficientNet-V2 Small, demonstrated exceptional efficiency by delivering nearly the same performance as its much larger Large version.

When considering the Transformer-based architectures, interesting results also emerged in this category. The standard ViT models performed at a tier below the top CNNs, showing moderate performance. The Swin Transformer family, which offers a hierarchical structure, was again the most successful among the Transformers, and the Swin-Base model ranked among the top performers alongside the best ResNets with an F1-Score of 90.43%. However, unlike in the 'Deck' class, the performance of the Swin-Large model decreased slightly compared to the Base model. The most striking result in this category belongs to the BEiT family. While the BEiT-Base model achieved a competitive result, the BEiT-Large model exhibited the worst performance among all models with a very low F1-Score of 75.10%. This dramatic drop demonstrates that the self-supervised pre-training strategy of BEiT, when combined with a large model capacity, failed to generalize on the visual characteristics of the 'Pavement' dataset and was entirely unsuccessful.

Evaluating the results not only based on absolute performance but also along the axis of efficiency reveals a very clear picture for the 'Pavement' class. The undisputed champions of this category, for both absolute performance and efficiency, are the ResNet-18 and ResNet-34 models. While ResNet-34 achieved the highest F1-Score,

ResNet-18 also presents an exceptional cost-performance profile by delivering nearly the same performance as the top models despite being one of the most efficient models. These findings strongly demonstrate that for practical applications involving crack detection on visually noisy and complex surfaces like pavements, the most sensible and optimal solution is to use proven, efficient, and medium-depth architectures like ResNet-18 or ResNet-34, rather than resorting to extremely large and complex models. The confusion matrices for the ResNet-34 and ResNet-18 models, which were the top performers for the 'Pavement' class, are presented in Figure 22.

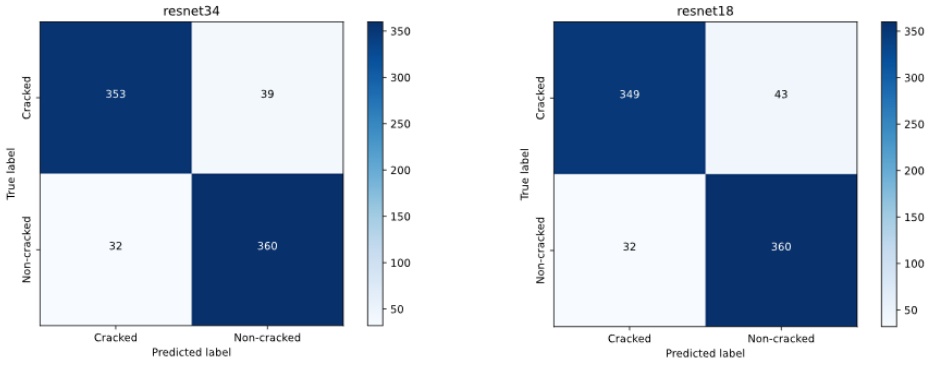

Figure 22. Confusion matrices for the two best-performing models on the 'Pavement' binary classification task: ResNet-34 (left) and ResNet-18 (right).

Table 4. Performance comparison of all evaluated architectures on the 'Wall' dataset for the binary classification task.

Models			Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Params	GFLOPs
ResNet	18		88.51	88.58	88.51	88.51	11.18	3.6470
	34		87.22	87.27	87.22	87.21	21.29	7.3565
	50		88.51	88.54	88.51	88.51	23.51	8.2634
	101		88.60	88.69	88.60	88.59	42.5	15.7288
	152		87.91	87.95	87.91	87.91	58.15	23.2038
DenseNet	121		87.31	87.39	87.31	87.30	6.96	5.6667
	169		84.89	85.10	84.89	84.86	12.49	6.7169
	201		83.85	84.01	83.85	83.83	18.1	8.5795
EfficientNet-V2	Small		88.95	89.01	88.95	88.94	20.18	5.4192
	Medium		87.39	87. 51	87.39	87.38	52.86	10.4436
	Large		87.74	87.77	87.74	87.74	117.24	23.9733
Vision Transformers	Patch-16	Tiny	84.28	84.32	84.28	84.28	5.52	2.1493
		Small	84.37	84.60	84.37	84.37	21.67	8.4817
		Base	84.80	85.02	84.80	84.78	85.8	33.6955
		Large	83.68	85.11	83.68	83.51	303.3	119.2923
	Patch-32	Small	80.57	80.68	80.57	80.55	22.49	2.2390
		Base	83.85	84.23	83.85	83.81	87.46	8.7247
		Large	83.85	84.04	83.85	83.83	305.51	30.5073

Swin Transformers	Tiny	86.96	87.08	86.96	86.95	27.52	8.7422
	Small	86.36	87.03	86.36	86.29	48.84	17.0885
	Base	87.13	87.27	87.13	87.12	86.75	30.3375
	Large	87.31	87.40	87.31	87.30	195.0	68.1649
Beit Transformers	Base	77.55	78.78	77.55	77.30	85.76	25.3294
	Large	86.70	86.80	86.70	86.69	303.41	89.5463

The results of the binary classification task conducted on the 'Wall' dataset, presented in detail in Table 4, depict a performance landscape that is distinct and unique from the other two categories. The most remarkable and important finding in this task is that the EfficientNet-V2 Small model, one of the most efficient members of the EfficientNet-V2 family, exhibited the highest performance among all tested architectures with an F1-Score of 88.94%. This result is strong evidence underscoring that the highest accuracy does not always come from the largest or most complex model. The ResNet family also demonstrated highly competitive and stable performance, with variants such as ResNet-101 and ResNet-18/50 achieving F1-scores above 88.5%. The most successful results for the "Wall" class appear to be obtained by modern and efficiency-focused CNN architectures.

When the performance within the CNN-based architectures is examined more deeply, it is clear that the winner of this category is the EfficientNet-V2 family. The fact that the higher-performing EfficientNet-V2 Small model was not surpassed by its larger Medium and Large variants suggests that the capacity and architectural design of the Small model are optimal for learning the visual features of 'Wall' surfaces (e.g., formwork lines, water stains). The ResNet family once again delivered very strong and reliable results. The performance of all

variants within the family being very close (in the 87.2% - 88.6% F1 range) confirms that the ResNet architecture provides a stable foundation for this task, even at different depths. In contrast, the DenseNet family, in a consistent trend observed in the previous two categories, exhibited weaker performance compared to other CNNs in this task as well, and its performance decreased as model depth increased. This situation reinforces the finding that scaling DenseNet did not provide an advantage for any of the three surface types in our project.

The Transformer-based architectures lagged slightly behind the top-performing members of the CNNs in the "Wall" category. The ViT and Swin Transformer families delivered upper-mid-range performance with F1-scores generally below 87%. In the Swin Transformer family, it is noteworthy that the performance gain from scaling the model size from Tiny to Large was quite marginal. One of the most interesting results in this category was observed in the BEiT family. In contrast to the failure of the BEiT-Large model on the other two datasets, in this task, BEiT-Large (86.69% F1) demonstrated a much superior performance compared to the BEiT-Base model (77.30% F1). Although this situation indicates that the features learned via BEiT's self-supervised pre-training strategy interacted better with a larger model capacity on the unique visual patterns of the 'Wall' dataset, this performance still did not reach the level of the best CNNs.

The result of this comprehensive analysis for the "Wall" class presents an exceptionally clear picture of the relationship between performance and efficiency. The undisputed winner of this category is the EfficientNet-V2 Small model, which not only achieved the highest F1-Score in terms of absolute performance but also accomplished this as one of the most efficient models. This model exhibits an outstanding cost-performance balance by delivering the highest accuracy at one of the lowest computational costs. Similarly, ResNet-18 also proved to be an excellent alternative for practical applications by achieving a result very close to the highest performance levels at a very low cost. In effect, for the detection of cracks on wall surfaces, this study has shown that using a

modern, efficient, and compact CNN architecture like EfficientNet-V2 Small offers the most optimal and effective solution, rather than resorting to extremely large and costly models. A detailed breakdown of the prediction distributions and error types for the two most successful architectures in this category, EfficientNet-V2 Small and ResNet-18, is illustrated by their respective confusion matrices in Figure 23.

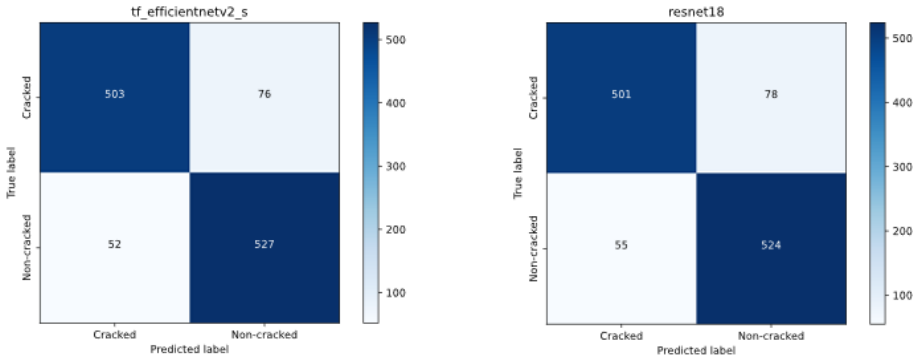

Figure 23. Confusion matrices for the two best-performing models on the 'Wall' binary classification task: EfficientNet-V2 Small (left) and ResNet-18 (right).

Table 5. Performance comparison of all evaluated architectures for the 6-class classification task on the unified dataset.

Models		Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Params	GFLOPs
ResNet	18	86.56	86.69	86.23	86.22	11.18	3.6470
	34	86.36	86.01	86.82	85.85	21.29	7.3565
	50	86.09	85.95	85.56	85.63	23.51	8.2634
	101	86.64	86.25	86.18	86.20	42.5	15.7288
	152	85.97	85.59	85.44	85.45	58.15	23.2038

DenseNet	121		84.44	83.98	83.89	83.86	6.96	5.6667
	169		82.76	82.89	82.19	82.12	12.49	6.7169
	201		82.48	82.29	81.66	81.73	18.1	8.5795
EfficientNet-V2	Small		87.11	86.60	86.45	86.42	20.18	5.4192
	Medium		86.72	86.51	86.24	86.24	52.86	10.4436
	Large		86.91	86.43	86.20	86.24	117.24	23.9733
Vision Transformers	Patch-16	Tiny	84.21	84.04	83.70	83.72	5.52	2.1493
		Small	83.70	83.49	83.21	83.20	21.67	8.4817
		Base	83.86	83.76	83.68	83.71	85.8	33.6955
		Large	84.84	85.20	84.58	84.48	303.3	119.2923
	Patch-32	Small	82.48	82.31	82.09	82.05	22.49	2.2390
		Base	83.07	84.09	82.68	82.62	87.46	8.7247
		Large	83.46	83.86	83.34	83.37	305.51	30.5073
Swin Transformers	Tiny		85.38	84.86	84.75	84.77	27.52	8.7422
	Small		85.34	84.65	84.79	84.68	48.84	17.0885
	Base		84.95	84.88	84.25	84.28	86.75	30.3375
	Large		86.13	86.10	85.62	85.63	195.0	68.1649
Beit Transformers	Base		83.89	83.66	83.49	83.40	85.76	25.3294
	Large		84.56	84.46	83.98	83.95	303.41	89.5463

The results of the second and most challenging phase of the experimental study, the six-class multi-class classification task, measure not only the models' ability to detect the presence of a defect but also their capacity to understand the structural context (Deck, Pavement, or Wall) in which it appears. The findings presented in Table 5 demonstrate that in this complex task, the EfficientNet-V2 and ResNet families established a clear superiority over the Transformer-based architectures. The most remarkable and important result of this experiment is that the model achieving the highest F1-Score (86.42%) among all tested architectures was EfficientNet-V2 Small, one of the most efficient members of its family. This finding strongly proves that the largest model is not always the best solution, even for the most complex tasks, and that a well-designed and more compact architecture can deliver superior performance.

The performance within the CNN-based architectures clearly reveals the winner of this task. The EfficientNet-V2 family, with all its variants scoring above an 86% F1-score, demonstrated that it is the most suitable architecture for this challenging task. Notably, the fact that the smallest member of the family, the Small model, achieved a marginally better result than its much larger Medium and Large versions confirms that the balance of capacity and efficiency in this model is optimal for this task. The ResNet family also, once again, delivered extremely stable and strong performance. While all ResNet variants achieved very similar results (in the 85.4% - 86.2% F1 range), the fact that the most efficient member, ResNet-18, was among the top models demonstrates how powerful and reliable this classic architecture remains. As in the previous experiments, the DenseNet family delivered lower performance in this task compared to other CNNs, reinforcing the observation that it is not a suitable option for the tasks within the scope of this study.

The Transformer-based architectures could not reach the performance level of the top CNNs in this multi-class classification task. The most successful family among the Transformers was the Swin Transformer, where performance consistently increased with model size. The largest member of the family, Swin-

Large, ranked at the top among Transformer models with an F1-Score of 85.63%. The standard ViT and BEiT families, on the other hand, achieved more modest results, recording F1-scores around 83%. In contrast to what was observed in some of the binary classification tasks, it is noteworthy that for this more complex multi-class task, the Large variants in both the ViT and BEiT families performed better than their Base counterparts. This indicates that the increased number of classes and complexity allowed these models to benefit somewhat from their higher capacity, but this benefit was still not sufficient to compete with the best CNNs.

The results obtained from this most comprehensive experiment of the study offer an extremely important takeaway for practical applications: efficiency does not mean sacrificing performance, even in complex tasks. The undisputed champions of this task are the EfficientNet-V2 Small and ResNet-18 models, which not only ranked among the top in absolute performance but also did so with exceptional computational efficiency. These two models achieved a higher accuracy rate than the largest Transformer models, which have hundreds of millions of parameters, but at one-tenth or less of the parameter and computational cost. This finding clearly demonstrates that for a multifaceted and realistic problem involving both crack detection and contextual surface recognition, modern and efficiency-focused CNN architectures, in their current form, offer a much more effective, practical, and optimal solution compared to Transformer-based approaches. For a more detailed view of the class-specific prediction accuracy and inter-class confusion for the two most successful architectures, the confusion matrices for EfficientNet-V2 Small and ResNet-18 are illustrated in Figure 24.

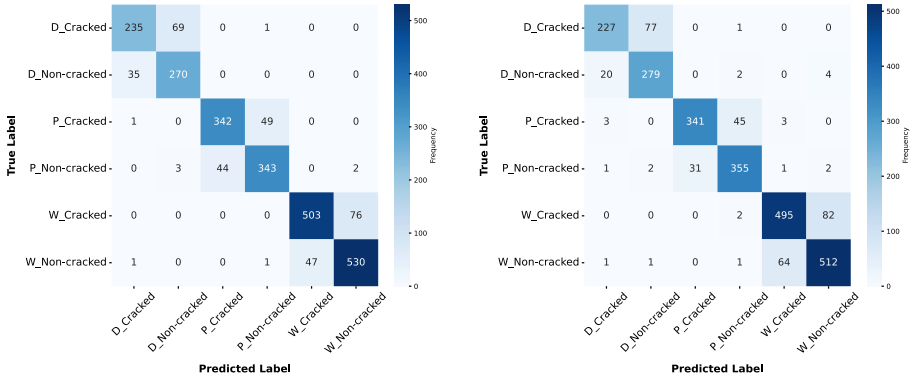

Figure 24. Confusion matrices for the two best-performing models on the unified 6-class classification task: EfficientNet-V2 Small (left) and ResNet-18 (right).

Discussion

The primary objective of this study was to conduct a comprehensive comparison of the performance of modern CNN and Transformer-based DL architectures in scenarios involving different surface types and task complexities for concrete crack detection. The experimental findings present a rich and multi-layered picture, revealing that no single architecture is universally superior across all tasks, but that certain architectural philosophies are more advantageous for specific tasks. The most fundamental and general finding is that efficiency-focused modern CNNs, namely EfficientNet-V2 and the classic ResNet family, exhibited more stable and generally higher performance compared to Transformer-based approaches in most of the tested tasks. Notably, one of the most remarkable results of this study is that smaller and more efficient models competed head-to-head with, and even surpassed, their counterparts that were many times larger and more complex.

The ResNet family, representing classic CNN architectures, depicted a highly reliable, stable, and high-performance profile throughout all experiments. The fact that the ResNet-34 model achieved the highest F1-Score among all tested architectures in the visually noisiest and most challenging category, "Pavement",

reveals the fundamental strength of this architecture. The success of ResNet lies in the ability to effectively train even very deep networks thanks to residual connections, and in the strong inductive bias afforded by its convolutional layers. This natural tendency to learn local spatial relationships (edges, textures) is perfectly suited for a task based on texture and pattern recognition like crack detection. Nevertheless, the observation that the deepest ResNet-152 model did not yield a better result than the medium-depth variants in the "Pavement" and "Deck" categories suggests that beyond a certain level of complexity, extreme depth may provide only a marginal contribution to performance or even be detrimental by starting to learn task-irrelevant noisy features.

In contrast to ResNet, another CNN family, DenseNet, underperformed without exception in all tasks within the scope of this study. In theory, the maximum feature reuse and strengthened gradient flow provided by dense connectivity have the potential to create highly efficient and powerful models. Indeed, in all three binary classification tasks, the best DenseNet performance was achieved by the smallest member of the family, DenseNet-121. However, the consistent drop in performance as model depth increased (DenseNet-169 and 201) indicates that there is an issue with the scalability of this architecture. A possible reason for this situation is that the concatenation of feature maps from all preceding layers in extremely deep DenseNet models might create a massive feature stack, causing the model to become confused about which features to focus on and complicating the optimization process. For the detection of fine details like cracks, this "feature inflation" may have brought more harm than good.

The star of the CNN wing in this study is undoubtedly the EfficientNet-V2 family. This architecture stood out not only for its high performance but also for delivering this performance with exceptional efficiency. The "compound scaling" philosophy at the core of EfficientNet, which is a strategy of increasing the network's depth, width, and resolution in a balanced manner, proved to be more intelligent and effective than brute-force approaches based solely on adding more

layers. The most striking result is that the EfficientNet-V2 Small model achieved the highest F1-Score in both the "Wall" and the most challenging unified six-class tasks. This is strong evidence that the highest accuracy does not always come from the largest model and that an efficiency-focused, intelligent design can be more important than raw computational power. The success of EfficientNet-V2 is a testament to how successful innovations like Fused-MBConv blocks, optimized for modern hardware, and progressive learning are at combining theoretical accuracy with practical efficiency.

Turning to the Transformer-based architectures, it was observed that the standard ViT generally could not reach the performance of the top CNNs. ViT's core philosophy of completely eliminating convolutional operations, and thus the locality bias, appears to be a disadvantage for this task. Since crack detection is by its nature based on the analysis of local texture and edge features, the built-in advantage that CNNs have in this regard outweighed ViT's ability to model global context. The fact that Patch-16 variants generally yielded better results than Patch-32 indicates that a higher-resolution view is necessary to capture the fine details of cracks, but even this was not enough to close the performance gap with the CNNs. The observation that the ViT-Large model performed worse than ViT-Tiny in the "Deck" task revealed that simply increasing the size of the ViT architecture is not an effective strategy for this task.

In contrast to the challenges faced by ViT, the Swin Transformer architecture stood out as the most successful member of the Transformer family. The secret to Swin's success is its intelligent reintroduction of two important CNN concepts that ViT abandoned: hierarchy and locality. By computing self-attention within local windows, which both increases computational efficiency (linear complexity), and by enabling inter-window communication through the shifted window mechanism, Swin effectively serves as a bridge between CNNs and Transformers. This hybrid approach allowed Swin to compete head-to-head with the best ResNet and EfficientNet models in tasks like "Deck" and "Pavement". The fact that Swin-Large showed the best performance in the "Deck" task, while

Swin-Base was a top performer in the "Pavement" task and the Large model could not surpass it, suggests that the optimal Swin size may vary depending on the visual characteristics of the task.

The BEiT models, which highlight the importance of pre-training strategies, yielded the most interesting and inconsistent results in this study. The complete failure of the BEiT-Large model on the "Pavement" dataset with a very low F1-Score of 75%, yet its significantly better performance than BEiT-Base on the "Wall" dataset, shows that the features obtained through self-supervised learning are extremely sensitive to the data distribution and visual nature of the fine-tuning task. The representations learned via Masked Image Modeling (MIM) might not have been generalizable to the weathered and noisy textures of "Pavement" surfaces, while they may have found a better match with the more structured and patterned features of "Wall" surfaces. This situation is an important finding that reveals how decisive not only the model architecture but also the type of data and task used in pre-training is on the final performance.

The two-phase structure of the experiments (binary vs. multi-class) also offers an important takeaway. As expected, when the task complexity increased, that is, in the six-class unified task, there was a drop in the absolute F1-scores of all models. However, the relative ranking of the model families was largely preserved: the EfficientNet-V2 and ResNet families were again at the top, while Swin Transformer followed them, and the other Transformers lagged further behind. This demonstrates that the models that are successful in the fundamental binary classification task also learn more powerful and discriminative features that allow them to handle the more complex task of contextual surface recognition. The fact that the EfficientNet-V2 Small model showed the highest performance even in this most challenging task is the ultimate proof of how superior its balance of efficiency and power is.

When all these results are brought together, an extremely clear roadmap for practical applications emerges. If the sole purpose in an application is to achieve the absolute highest accuracy and computational resources are not a constraint,

models like Swin-Large or EfficientNet-V2 Large could be preferred. However, the marginal performance increase provided by these models comes at the cost of a much higher parameter count and slow inference times. In contrast, this study demonstrates with overwhelming evidence that for almost all practical scenarios, the most optimal solution is efficient models. Specifically, models like EfficientNet-V2 Small and ResNet-18/34 delivered better performance than the top-performing models, at one-fifth or less of the computational cost. This situation clearly reveals that in real-world applications where resources are limited, fast inference times are necessary, and energy efficiency is important, these light and efficient CNN architectures are much more sensible and effective alternatives.

Lastly, it is important to acknowledge some limitations of this study and to indicate potential paths for future research. Although SDNET2018 is a comprehensive dataset, it would be beneficial to test these models on other datasets collected from different geographical regions and different types of concrete to verify the generalizability of the findings. This study focused on the task of classification; in future work, the best-performing backbone architectures (e.g., EfficientNet-V2 Small) could be combined with decoder structures like U-Net or DeepLab and be evaluated for semantic segmentation tasks, which determine the exact location and width of cracks. In addition, the effects of newer self-supervised learning methods or different pre-training strategies on this task could be investigated. Finally, further optimizing even the most efficient models with techniques such as model quantization and pruning so that they can run on mobile or embedded systems could be a practical extension of this research.

Conclusions

This comprehensive investigation aimed to present a comparative analysis of modern DL architectures, spanning both CNNs and Transformer-based models, to determine which architectural philosophies are most suitable for the specific and practical engineering problem of concrete crack detection. For this purpose,

a wide array of models from seven distinct architectural families (ResNet, DenseNet, EfficientNet-V2, ViT, Swin, BEiT) was meticulously tested on the SDNET2018 dataset across three different structural categories (Deck, Pavement, Wall) in both binary and multi-class classification tasks. The obtained experimental results present a rich and multi-layered picture, revealing that while no single model was universally superior across all scenarios, the general trend indicates that efficiency-focused modern CNNs, particularly EfficientNet-V2 and the classic ResNet family, exhibited more stable, reliable, and generally higher performance compared to Transformer-based approaches. The most fundamental finding of this study is the fact that achieving the highest accuracy does not always require the largest or most complex model; on the contrary, intelligently designed, more compact architectures can deliver the best results even in the most challenging tasks. In this context, the classic ResNet family once again proved the power and durability of the fundamental residual connection concept, with the medium-depth ResNet-34 variant achieving the highest performance in the visually noisiest and most challenging "Pavement" category. On the other hand, the consistent observation across all experiments of performance degradation with increasing model depth in the DenseNet architecture was a significant finding, indicating an issue with the scalability of this architecture for these tasks. The undisputed star on the CNN side was the EfficientNet-V2 family, which combines efficiency and performance through its "compound scaling" philosophy. The most striking result of our work is that the smallest member of the family, the EfficientNet-V2 Small model, achieved the highest F1-Score in both the "Wall" binary classification task and the most demanding six-class task, which required the models to recognize both the crack and the surface type simultaneously. This situation underscores that an architecture that intelligently balances parameter count and computational cost can be more effective than raw processing power. On the Transformer front, it was observed that Swin Transformer, which offers a hierarchical structure and a local attention mechanism similar to CNNs, was the most successful among the Transformers.

The standard ViT, lacking a strong locality bias, lagged behind the CNNs in this task, which is critical for texture and edge analysis. The inconsistent performance exhibited by the BEiT models across different datasets demonstrated how sensitive the features obtained through self-supervised learning are to the visual nature of the fine-tuning task. Ultimately, the clearest message this study offers for practical engineering applications is the following: in real-world scenarios where a balance between reliability, speed, and accuracy is required, the most optimal solution is compact CNN architectures with proven efficiency. Models like EfficientNet-V2 Small and ResNet-18 delivered higher performance metrics than the largest Transformer models with hundreds of millions of parameters, achieving this at one-tenth or less of the computational cost and thus offering an exceptional cost-performance ratio. This finding provides a concrete and applicable roadmap for the development of fast and reliable crack detection systems that can be integrated into autonomous inspection systems, drones, or mobile devices. Future work holds the potential to further advance success in this area by adapting these most successful architectures to segmentation tasks and by developing more task-specific pre-training strategies.

REFERENCES

[1] T. Huang, C. Wan, T. Liu, C. Miao, Degradation law of bond strength of reinforced concrete with corrosion-induced cracks and machine learning prediction model, Journal of Building Engineering 98 (2024) 111022. https://doi.org/10.1016/J.JOBE.2024.111022.

[2] J. Pouya, M. Neji, L. De Windt, F. Péralès, A. Socié, J. Corvisier, Investigating chemical and cracking processes in cement paste exposed to a low external sulfate attack with emphasis on the contribution of gypsum, Constr Build Mater 413 (2024) 134845. https://doi.org/10.1016/J.CONBUILDMAT.2023.134845.

[3] H. Zeng, M. Jin, W. Li, C. Gao, Y. Ma, Q. Guan, J. Liu, Performance evolution of low heat cement under thermal cycling fatigue: A comparative study with moderate heat cement and ordinary Portland cement, Constr Build Mater 412 (2024) 134863. https://doi.org/10.1016/J.CONBUILDMAT.2024.134863.

[4] T. Feng, Y. Miao, Y. Tan, Z. Yang, T. Cao, F. Wang, J. Jiang, Prediction Methodology for the Service Life of Concrete Structures in Marine Environment: From Materials to Performance, Engineering (2025). https://doi.org/10.1016/J.ENG.2025.03.010.

[5] S. Ullmann, D. Lowke, The effect of external load on the chloride migration resistance and the service life of reinforced concrete structures and repair mortars, Constr Build Mater 443 (2024) 137770. https://doi.org/10.1016/J.CONBUILDMAT.2024.137770.

[6] M. Sohaib, M.J. Hasan, M.A. Shah, Z. Zheng, A robust self-supervised approach for fine-grained crack detection in concrete structures, Scientific Reports 2024 14:1 14 (2024) 1–20. https://doi.org/10.1038/s41598-024-63575-x.

[7] Q.; Yuan, Y.; Shi, M.A. Li, M. Libera Battagliere, V. Gagliardi, Q. Yuan, Y. Shi, M. Li, A Review of Computer Vision-Based Crack Detection Methods in Civil Infrastructure: Progress and Challenges, Remote

Sensing 2024, Vol. 16, Page 2910 16 (2024) 2910. https://doi.org/10.3390/RS16162910.

[8] A.N. Beskopylny, S.A. Stel'makh, E.M. Shcherban', I. Razveeva, A. Kozhakin, B. Meskhi, A. Chernil'nik, D. Elshaeva, O. Ananova, M. Girya, T. Nurkhabinov, N. Beskopylny, Computer Vision Method for Automatic Detection of Microstructure Defects of Concrete, Sensors 2024, Vol. 24, Page 4373 24 (2024) 4373. https://doi.org/10.3390/S24134373.

[9] H. Kaveh, R. Alhajj, Recent advances in crack detection technologies for structures: a survey of 2022-2023 literature, Front Built Environ 10 (2024) 1321634. https://doi.org/10.3389/FBUIL.2024.1321634/XML/NLM.

[10] Y. Cakmak, S. Safak, M.A. Bayram, I. Pacal, Comprehensive Evaluation of Machine Learning and ANN Models for Breast Cancer Detection, Computer and Decision Making: An International Journal 1 (2024) 84–102. https://doi.org/10.59543/COMDEM.V1I.10349.

[11] Y. Cakmak, I. Pacal, Enhancing Breast Cancer Diagnosis: A Comparative Evaluation of Machine Learning Algorithms Using the Wisconsin Dataset, Journal of Operations Intelligence 3 (2025) 175–196. https://doi.org/10.31181/JOPI31202539.

[12] Pacal Ishak, Cakmak Yigitcan, DIAGNOSTIC ANALYSIS OF VARIOUS CANCER TYPES WITH ARTIFICIAL INTELLIGENCE, 2025. www.duvaryayinlari.com.

[13] J. Zeynalov, Y. Çakmak, İ. Paçal, Automated Apple Leaf Disease Classification Using Deep Convolutional Neural Networks: A Comparative Study on the Plant Village Dataset, Journal of Computer Science and Digital Technologies 1 (2025) 5–17. https://doi.org/10.61640/jcsdt.2025.0601.

[14] Y. Qi, Z. Ding, Y. Luo, Z. Ma, A Three-Step Computer Vision-Based Framework for Concrete Crack Detection and Dimensions Identification,

Buildings 2024, Vol. 14, Page 2360 14 (2024) 2360. https://doi.org/10.3390/BUILDINGS14082360.

[15] J. Liu, H. Sun, H. Liu, Q. Yue, Z. Xu, Y. Jia, S. Wang, Recognition and quantification of apparent damage to concrete structure based on computer vision, Measurement 240 (2025) 115635. https://doi.org/10.1016/J.MEASUREMENT.2024.115635.

[16] I. Pacal, O. Akhan, R.T. Deveci, M. Deveci, NeXtBrain: Combining local and global feature learning for brain tumor classification, Brain Res 1863 (2025) 149762. https://doi.org/10.1016/J.BRAINRES.2025.149762.

[17] S. Ince, I. Kunduracioglu, B. Bayram, I. Pacal, U-Net-Based Models for Precise Brain Stroke Segmentation, Chaos Theory and Applications 7 (2025) 50–60. https://doi.org/10.51537/CHAOS.1605529.

[18] B. Bayram, I. Kunduracioglu, S. Ince, I. Pacal, A systematic review of deep learning in MRI-based cerebral vascular occlusion-based brain diseases, Neuroscience 568 (2025) 76–94. https://doi.org/10.1016/J.NEUROSCIENCE.2025.01.020.

[19] I. Pacal, O. Attallah, InceptionNeXt-Transformer: A novel multi-scale deep feature learning architecture for multimodal breast cancer diagnosis, Biomed Signal Process Control 110 (2025) 108116. https://doi.org/10.1016/J.BSPC.2025.108116.

[20] B. Ozdemir, E. Aslan, I. Pacal, Attention Enhanced InceptionNeXt Based Hybrid Deep Learning Model for Lung Cancer Detection, IEEE Access (2025). https://doi.org/10.1109/ACCESS.2025.3539122.

[21] M.A.H. Lubbad, I.L. Kurtulus, D. Karaboga, K. Kilic, A. Basturk, B. Akay, O.U. Nalbantoglu, O.M.D. Yilmaz, M. Ayata, S. Yilmaz, I. Pacal, A Comparative Analysis of Deep Learning-Based Approaches for Classifying Dental Implants Decision Support System, Journal of Imaging Informatics in Medicine 37 (2024) 2559–2580. https://doi.org/10.1007/S10278-024-01086-X/FIGURES/14.

[22] I. Leblebicioglu Kurtulus, M. Lubbad, O.M.D. Yilmaz, K. Kilic, D. Karaboga, A. Basturk, B. Akay, U. Nalbantoglu, S. Yilmaz, M. Ayata, I. Pacal, A robust deep learning model for the classification of dental implant brands, J Stomatol Oral Maxillofac Surg 125 (2024) 101818. https://doi.org/10.1016/J.JORMAS.2024.101818.

[23] M. Lubbad, D. Karaboga, A. Basturk, B. Akay, U. Nalbantoglu, I. Pacal, Machine learning applications in detection and diagnosis of urology cancers: a systematic literature review, Neural Comput Appl 36 (2024) 6355–6379. https://doi.org/10.1007/S00521-023-09375-2/TABLES/6.

[24] P. Panwar, K. Goyal, J.K. Shandilya, Deep Learning-Enabled Health Assessment for Sustainable Maintenance of Existing Concrete Structures: A Review, Springer Tracts in Civil Engineering Part F219 (2025) 93–121. https://doi.org/10.1007/978-981-97-8975-7_3.

[25] R. Kirthiga, S. Elavenil, A survey on crack detection in concrete surface using image processing and machine learning, Journal of Building Pathology and Rehabilitation 2023 9:1 9 (2023) 1–25. https://doi.org/10.1007/S41024-023-00371-6.

[26] V. Pandey, S. Sharan Mishra, A review of image-based deep learning methods for crack detection, Multimedia Tools and Applications 2025 (2025) 1–43. https://doi.org/10.1007/S11042-025-20729-X.

[27] L.M. Arpitha, R.A. Kumar, Z. Fathima, R. Yeshaswini, G. Dhanyashree, D. Roshini, Comprehensive Analysis of Machine Learning Techniques for Crack Detection, Sustainable Civil Infrastructures (2025) 169–192. https://doi.org/10.1007/978-3-031-83750-0_12.

[28] Navpreet, R.K. Roul, R. Rani, Comparative Analysis of Machine Learning and Deep Learning Classifiers for Crack Classification, Lecture Notes in Networks and Systems 1085 LNNS (2024) 191–203. https://doi.org/10.1007/978-981-97-6726-7_15.

[29] S.G.A. Usha, Cutting-Edge Network Based Concrete Crack Detection and Analysis for Structural Health Monitoring, Springer Tracts in Civil

Engineering Part F219 (2025) 157–175. https://doi.org/10.1007/978-981-97-8975-7_5.

[30] Y.M. Abbas, H. Alghamdi, Semantic segmentation and deep CNN learning vision-based crack recognition system for concrete surfaces: development and implementation, Signal Image Video Process 19 (2025) 1–15. https://doi.org/10.1007/S11760-025-03913-2/FIGURES/18.

[31] S.K. Bussa, N.K. Boppana, Enhanced ResNet50 deep learning algorithm for classification of crack images in RCC structures, Asian Journal of Civil Engineering (2025) 1–12. https://doi.org/10.1007/S42107-025-01396-7/TABLES/3.

[32] Q. Dai, M. Ishfaque, S.U.R. Khan, Y.L. Luo, Y. Lei, B. Zhang, W. Zhou, Image classification for sub-surface crack identification in concrete dam based on borehole CCTV images using deep dense hybrid model, Stochastic Environmental Research and Risk Assessment (2024) 1–18. https://doi.org/10.1007/S00477-024-02743-X/FIGURES/14.

[33] J. Wang, T. Ueda, P. Wang, Z. Li, Y. Li, Building damage inspection method using UAV-based data acquisition and deep learning-based crack detection, J Civ Struct Health Monit 15 (2024) 151–171. https://doi.org/10.1007/S13349-024-00836-3/FIGURES/11.

[34] Structural Defects Network (SDNET) 2018, (n.d.). https://www.kaggle.com/datasets/aniruddhsharma/structural-defects-network-concrete-crack-images (accessed July 1, 2025).

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

[36] M. JGMv, Fracture processes of concrete: assessment of material parameters for fracture models, (1997).

[37] H. Ahmed, A. Zahedi, L.F.M. Sanchez, P.L. Fecteau, Condition assessment of ASR-affected reinforced concrete columns after nearly 20

years in service, Constr Build Mater 347 (2022) 128570. https://doi.org/10.1016/J.CONBUILDMAT.2022.128570.

[38] M.G. Richardson, Fundamentals of Durable Reinforced Concrete, Fundamentals of Durable Reinforced Concrete (2023) 1–390. https://doi.org/10.1201/9781003261414/FUNDAMENTALS-DURABLE-REINFORCED-CONCRETE-MARK-RICHARDSON/RIGHTS-AND-PERMISSIONS.

[39] J. Jakubowski, K. Tomczak, Deep learning metasensor for crack-width assessment and self-healing evaluation in concrete, Constr Build Mater 422 (2024) 135768. https://doi.org/10.1016/J.CONBUILDMAT.2024.135768.

[40] K.B. Shahrbijari, J.A.O. Barros, I.B. Valente, Experimental study on the structural performance of concrete beams reinforced with prestressed GFRP and steel bars, Constr Build Mater 438 (2024) 137031. https://doi.org/10.1016/J.CONBUILDMAT.2024.137031.

[41] A. Borosnyói, G.L. Balázs, Models for flexural cracking in concrete: the state of the art, Structural Concrete 6 (2005) 53–62. https://doi.org/10.1680/STCO.2005.6.2.53.

[42] A. Codina, L. Torres, T. D'Antino, M. Baena, C. Barris, Flexural performance of RC beams strengthened with HB CFRP plates: Experimental study and theoretical model based on the intermediate crack debonding, Constr Build Mater 458 (2025) 139444. https://doi.org/10.1016/J.CONBUILDMAT.2024.139444.

[43] E.M. Silva, K.A. Harries, P. Ludvig, S. Sólyom, S. Platt, Experimental investigation of bond and cracking behaviours in gfrp-reinforced concrete members, Journal of Building Engineering 83 (2024) 108434. https://doi.org/10.1016/J.JOBE.2024.108434.

[44] V. Picandet, A. Khelidj, H. Bellegou, Crack effects on gas and water permeability of concretes, Cem Concr Res 39 (2009) 537–547. https://doi.org/10.1016/J.CEMCONRES.2009.03.009.

[45] L. Mengel, H.W. Krauss, D. Lowke, Water transport through cracks in plain and reinforced concrete – Influencing factors and open questions, Constr Build Mater 254 (2020) 118990. https://doi.org/10.1016/J.CONBUILDMAT.2020.118990.

[46] R. Saliger, High Grade Steel in Reinforced Concrete, in: Proceedings of the 2nd Congress of the International Association for Bridge and Structural Engineering (IABSE), IABSE Publications, Berlin-Munich, 1936: pp. 293–315.

[47] N.J. Carino, J.R. Clifton, A. Prabhakar, Prediction of Cracking in Reinforced Concrete Structures, 1995.

[48] B.B. Broms, Crack Width and Crack Spacing In Reinforced Concrete Members, Journal Proceedings 62 (1965) 1237–1256. https://doi.org/10.14359/7742.

[49] J. Ferry-Borges, Cracking and deformability of reinforced concrete beams, Association International Des Ponts et Charpenters 26 (1966).

[50] T.P. Doğan, H. Kalkan, Ö. Aldemir, M. Ayhan, M. Böcek, Ö. Anıl, Investigation of RC structure damages after February 6, 2023, Kahramanmaraş earthquake in the Hatay region, Bulletin of Earthquake Engineering 22 (2024) 5201–5229. https://doi.org/10.1007/S10518-024-01965-2/TABLES/3.

[51] C. Européen, Eurocode 2: Design of concrete structures—Part 1-1: General rules and rules for buildings, London: British Standard Institution (2004) 37.

[52] C.E.N.F. 1992-1-1, Eurocode 2–design of concrete structures: part 1–1: general rules and rules for buildings, bridges and civil engineering structures, (2023).

[53] F.I. Du Béton, fib model code for concrete structures 2010, Wiley-vch Verlag Gmbh, 2013.

[54] F.I. Du Béton, fib model code for concrete structures 2010, Wiley-vch Verlag Gmbh, 2013.

[55] D. Borosnyoi-Crawley, Change of crack widths and anatomy of cracks within the cover of reinforced concrete tension members, Constr Build Mater 489 (2025) 142192. https://doi.org/10.1016/J.CONBUILDMAT.2025.142192.

[56] İ. Feyza, Ç. Sdu, BETONARME BİNALARDA GÖZLENEN HASARLAR, NEDENLERİ VE ÖNERİLER, International Journal of Technological Sciences 3 (2011) 62–71. https://dergipark.org.tr/en/pub/utbd/issue/25986/273729 (accessed July 24, 2025).

[57] A. Mertol, C. Mertol, Deprem Mühendisliği, Depreme Dayankl Yap Tasarm, Kozan Ofset, Ankara (2002).

[58] C. Beklen, İ.H. Çağatay, Çerçevelerde Dolgu Duvar Modellerinin İncelenmesi, Çukurova Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi 24 (2016). https://dergipark.org.tr/tr/pub/cukurovaummfd/issue/22771/243029 (accessed July 24, 2025).

[59] F. Şermet, E. Ercan, E. Hökelekli, A. Demir, B. Arısoy, The behavior of concrete-encased steel composite column-beam joints under cyclic loading, The Structural Design of Tall and Special Buildings 30 (2021) e1842. https://doi.org/10.1002/TAL.1842.

[60] F. Şermet, E. Ercan, E. Hökelekli, B. Arisoy, Cyclic behavior of composite column-reinforced concrete beam joints, Sigma Journal of Engineering and Natural Sciences 38 (2021) 1427–1445. https://dergipark.org.tr/en/pub/sigma/issue/65286/1007464 (accessed July 24, 2025).

[61] Z. Wang, P. Wang, K. Liu, P. Wang, Y. Fu, C.-T. Lu, C.C. Aggarwal, J. Pei, Y. Zhou, A Comprehensive Survey on Data Augmentation, (2024). https://arxiv.org/abs/2405.09591v3 (accessed May 28, 2025).

[62] A. Mumuni, F. Mumuni, N.K. Gerrar, A Survey of Synthetic Data Augmentation Methods in Machine Vision, Machine Intelligence

Research 2024 21:5 21 (2024) 831–869. https://doi.org/10.1007/S11633-022-1411-7.

[63] G. Mesnil, Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I. Goodfellow, E. Lavoie, X. Muller, G. Desjardins, D. Warde-Farley, P. Vincent, A. Courville, J. Bergstra, Unsupervised and Transfer Learning Challenge: a Deep Learning Approach, 27 (2012) 97–110. https://proceedings.mlr.press/v27/mesnil12a.html (accessed July 1, 2025).

[64] Y. Bengio, Deep Learning of Representations for Unsupervised and Transfer Learning, 27 (2012) 17–36. https://proceedings.mlr.press/v27/bengio12a.html (accessed July 1, 2025).

[65] M. Iman, H.R. Arabnia, K. Rasheed, A Review of Deep Transfer Learning and Recent Advancements, Technologies 2023, Vol. 11, Page 40 11 (2023) 40. https://doi.org/10.3390/TECHNOLOGIES11020040.

[66] Z. Ren, H. Zhang, T. Huang, D. Yang, W. Zhang, Y. Jiang, Y. Zhou, X. Zhang, Y. Wang, J. Gupta, S. Pathak, G. Kumar, Deep Learning (CNN) and Transfer Learning: A Review, J Phys Conf Ser 2273 (2022) 012029. https://doi.org/10.1088/1742-6596/2273/1/012029.

[67] Z. Zhao, L. Alzubaidi, J. Zhang, Y. Duan, Y. Gu, A comparison review of transfer learning and self-supervised learning: Definitions, applications, advantages and limitations, Expert Syst Appl 242 (2024) 122807. https://doi.org/10.1016/J.ESWA.2023.122807.

[68] Deep Learning and Transfer Learning Approaches for Image Classification, (2019). https://www.researchgate.net/publication/333666150 (accessed July 1, 2025).

[69] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, C. Liu, A Survey on Deep Transfer Learning, Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in

Bioinformatics) 11141 LNCS (2018) 270–279. https://doi.org/10.1007/978-3-030-01424-7_27.

[70] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, (n.d.). http://image-net.org/challenges/LSVRC/2015/ (accessed July 1, 2025).

[71] G. Huang, Z. Liu, L. van der Maaten, K.Q. Weinberger, Densely Connected Convolutional Networks, (2018). http://arxiv.org/abs/1608.06993.

[72] M. Tan, Q. V. Le, EfficientNetV2: Smaller Models and Faster Training, (2021). http://arxiv.org/abs/2104.00298.

[73] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE, (n.d.). https://github.com/ (accessed July 1, 2025).

[74] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, (n.d.). https://github. (accessed July 1, 2025).

[75] H. Bao, L. Dong, S. Piao, F. Wei, BEIT: BERT Pre-Training of Image Transformers, (n.d.). https://aka.ms/beit (accessed July 1, 2025).