Received: December 30, 2025. Revised: February 25, 2026. 988

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Transformer Model Coupled with Dilated Convolution for Improved Crack Detection

Joseph James[1]* **Lipsa Nayak[1]**

[1] *Department of Computer Science-PG, Vels Institute of Science, Technology and Advance Studies VISTAS, Chennai-177, India*
* Corresponding author's Email: josephjames444jp@gmail.com

Abstract: Structural health monitoring is one of the common and challenging issues in civil engineering. Among the different factors affecting the health of the structure, Crack plays a major role. The identification of cracks is not easy but the early detection of cracks can avoid unpleasant situations in real time. But it exists as a challenge for engineers. This study offers a novel strategy that makes use of deep learning techniques. To get accuracy, the suggested model combines fully linked networks, feature fusion, and feature extraction. To get the best result for feature extraction, a preprocessing module, a dilated convolution network, and a vision transformer are utilized. The model is tested on a dataset that included 40,000 labels labelled" crack and non-crack," and the findings indicate that the suggested model outperforms the state-of-the-art techniques in terms of accuracy, achieving 98.05%. The deep learning technology can help overcome the shortcomings of the most advanced method for categorizing cracks on concrete surfaces, in addition to producing an accurate result.

Keywords: Crack, Detection, Vision transformers, Dilated convolution network, Structural health monitoring (SHM), Feature fusion.

1. Introduction

In the past, most concepts existed only as theoretical models, causing limitations to technical advancements. This makes problem solving a hard task requiring more human potential to solve. Development in computer science leads to the development of a system with human-like intelligence capable of real-time problem-solving skills. AI has significantly impacted various domains, including civil engineering [1], where it enhances design, construction, maintenance, and management processes. In civil engineering, monitoring the safety, reliability and life span of the structure is a crucial task. SHM involves assessing structural integrity using sensors, data acquisition systems, and analytical techniques to detect and address issues like cracks, corrosion, and fatigue [2]. It provides insights into structural behaviour over time, enabling proactive maintenance that enhances safety, reduces downtime, and minimizes repair costs. Technological advancements in sensors, wireless communication, and data analytics have made SHM systems more sophisticated, supporting real-time monitoring and predictive maintenance. Among various structural issues, cracks are particularly critical due to their potential to compromise structural integrity and safety. Cracks can lead to moisture penetration, corrosion, and further deterioration. Detecting cracks promptly is essential for addressing their root causes and implementing effective remedial measures. Early crack detection allows engineers to take timely actions, avoiding catastrophic failures and ensuring cost-effective maintenance.

Crack detection in SHM serves several purposes. It acts as an early warning system for structural distress, facilitating proactive interventions. Additionally, monitoring crack size, location, and propagation provides valuable data for assessing structural health and performance, guiding maintenance schedules and operational decisions.

Crack detection also supports condition-based maintenance, optimizing resource allocation and extending the service life of infrastructure. Various crack detection techniques exist, ranging from traditional manual methods to advanced automated approaches. Deep learning has emerged as a prominent tool for crack detection due to its superior performance in handling complex data[2]. Neural networks, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), excel in tasks like image recognition and sequential data analysis. CNNs effectively extract spatial features from images, making them suitable for detecting cracks in concrete structures [3]. Moreover, deep learning models automate feature learning, eliminating the need for manual engineering and enabling scalability for large- scale projects. Despite its advantages, deep learning requires extensive training data and computational resources. Ensuring the generalization and robustness of models across different concrete structures remains a challenge. Nevertheless, its capabilities for automation and efficient data handling motivate ongoing research in applying deep learning to crack detection. By integrating predictive modeling and data analytics, deep learning offers transformative potential for SHM, ensuring the reliability, safety, and sustainability of infrastructure systems.The research paper is structured as: a thorough analysis of current crack detection techniques given in Section 2, which high- lights their advantages and disadvantages; the suggested deep learning-based crack detection framework is presented in Section 3, which describes the model's training and assessment approach; the conclusion is outlined in Section 4, which addresses the benefits of the suggested strategy over cutting-edge techniques and suggests possible avenues for further study.

1.1 Problem definition

Despite significant progress in vision-based crack detection, existing approaches still suffer from notable limitations that restrict their reliability in real-world Structural Health Monitoring (SHM) applications. Traditional image-processing methods rely heavily on handcrafted features and threshold-based edge detection, making them sensitive to noise, illumination variation, and surface texture. Conventional CNN-based models, although effective in local feature extraction, often fail to capture long-range contextual dependencies and multi-scale crack patterns, particularly for thin, irregular, or discontinuous cracks. Furthermore, deeper CNN architectures tend to suffer from high computational complexity, vanishing gradient issues, and saturation in performance gains.

Recent transformer-based approaches address global dependency modeling but struggle with local spatial feature extraction and typically require large-scale datasets and high computational resources. Therefore, there exists a research gap in designing a crack detection framework that can simultaneously capture global contextual relationships and fine-grained local features while maintaining high accuracy and computational efficiency.

To address these challenges, this work proposes a hybrid architecture that integrates Vision Transformers (ViT) with Dilated Convolutional Neural Networks (DCNN). The proposed model leverages the global self-attention capability of ViTs and the multi-scale receptive field of dilated convolutions, enabling robust crack classification under diverse environmental conditions. This integration clearly differentiates the proposed work from existing single-model and purely CNN-based approaches.

2. Literature review

This section provides a detailed review of different computational methods and models discussed in recent articles published from 2019 to 2023. It helps to understand the current state of the art in terms of architecture, accuracy and limitations in structural health monitoring. CNNs are specifically designed to capture spatial hierarchies in image data efficiently. Their architecture excels at learning local features like edges, textures, and patterns. Pre-trained models are powerful and time-saving but lack fine-tuned adaptability for specialized tasks, making CNNs a preferred choice in classification experiments. As a result, most of the CNN-based models and pre-trained models used in the literature show above 90% accuracy. It shows CNN-based models and pre-trained models better in classification while considering the number of literatures as a parameter for most of the experiments which use CNN for the classification. In [4] Qianyun Zhang uses an integration of a one-dimensional convolutional neural network (1D-CNN) where a long short-term memory (LSTM) 1D-CNN-LSTM algorithm is used to develop a vision-based crack detector. In addition, the images were transforms in to flattened frequency data in the preprocessing phase to improve the performance of the model. In contrast, Young-Jin [5] uses CNN to develop a crack detector and the comparison of the proposed method with traditional edge detection methods (i.e., Canny and Sobel), which also states that the proposed method

shows very robust performance and the Sobel and Canny edge detection methods provided no meaningful crack information. In [6] Laura Falaschetti uses two lightweight CNN models for the implementation. Apart from another vision-based model, Laura Falaschetti proposes a low-cost, low-power platform to detect concrete cracks in real time named Open MV Cam H7 Plus. In[7] Sharmad Bhat proposes a method which identifies cracks and non-cracks and also can classifies the type of crack into longitudinal crack or not. For the architecture, Sharmad Bhat uses 3 convolutional layers with 3 max-pooling layers along with a flattened layer and two dense layers. For an input, greyscale conversion and resizing take place as pre-processing and reactivation functions. Output is taken using the sigmoid activation function. Multi-scale dilated convolution module with a novel characterization algorithm is proposed by Weidong Song[8] to classify the crack after the detection of crack and the proposed model classify the crack into four types: transversal, longitudinal, block, and alligator and this is better than the model proposed by Sharmad Bhat, which classify longitudinal crack only. In [9] Dimitris Dais uses transfer learning along with CNN for the classification and segmentation but Suguru Yokoyama uses CNN and the difference is that one classifier is to detect cracks on masonry surfaces and another for concrete surfaces. Keqin Chen[10] and Hyunjun Kim[11] use CNN with some additional methods and algorithms to enhance the performance. Keqin Chen uses the Adam optimization algorithm and batch normalization (BN) algorithm, and Hyunjun Kim, proposes the crack candidate region (CCR) framework and the experiments conclude that the CNN method shows better accuracy than the SURF-based method. It shows the performance of CNN over the other methods. For the clear identification of crack Young Choi uses clustering method along with convolutional neural network to achieve better results. Xuwei Dong proposes an improved lightweight algorithm, YOLOv8-Crack Detection (YOLOv8-CD), based on YOLOv8. Visual attention networks (VANs) and large con volutional attention (LCA) modules are used to leverage the algorithm's strength. In addition, a large separable kernel attention (LSKA) module is used to extract features of concrete surface cracks [12]Young Choi uses the clustering technique and convolutional neural network to detect the crack [13]. Apart from all the methods used in the literature, Georgiana-Lucia Coca [14] proposes an approach based on CNN with sixteen iterations and it shows higher accuracy with high time complexity as a disadvantage. Among the relevant literature, Georgiana- Lucia Coca stands apart due to the convolutional neural network (CNN) algorithm with sixteen iterations. The rest of the literature uses CNN itself but one of the works of literature mentioned uses two lightweight CNNs[5] to attain the result. Bubryur Kim [15] proposed LeNet-5, a CNN architecture for the detection which is experimented on Middle East Technical University (METU) dataset and the model attained an accuracy of 99.8% with minimum computation. In [16] Mustafa Abubakr compared two CNN model named Xceptiion and Vanilla, which shows Xception model attain 9.78% accuracy than Vanilla model. ImageNet pre-trained VGG-16 DCNN were used for the classification[17], Kasthuriran- gan Gopalakrishnan stated that the proposed model achieved 90% accuracy. In addition, the model is capable to classify the crack in real time without any pre-processing and augmentation. CaNet based architecture were proposed with ResNet50 as backbone which shows 10.2% less accuracy than LeNet-5.

2.1 Drawbacks of conventional techniques

1. Traditional Image Processing Methods

Traditional crack detection techniques such as Sobel, Canny, and threshold-based edge detection depend on handcrafted features.

Drawbacks:

- Highly sensitive to noise, shadows, and lighting variations
- Poor performance on complex backgrounds and textured concrete surfaces
- Inability to generalize across different crack shapes and scales
- Require manual parameter tuning

2. Classical Machine Learning Methods

Methods using SVM, k-means clustering, and SURF-based features improve automation but still depend on manually designed features.

Drawbacks:

- Feature engineering is time-consuming and domain-dependent
- Limited scalability for large datasets
- Reduced robustness when applied to diverse crack patterns

3. CNN-Based Approaches

CNNs dominate recent crack detection research due to their automatic feature learning capability.

Drawbacks:

- Limited receptive field restricts long-range dependency modeling
- Deeper CNNs increase computational cost and risk vanishing gradients
- Pooling operations may discard fine crack details
- Performance saturation despite architectural depth increase

4. Lightweight and Transfer Learning Models

Lightweight CNNs and transfer learning reduce training time.

Drawbacks:

- Reduced representational capacity
- Pre-trained features may not optimally adapt to crack-specific patterns
- Accuracy trade-offs compared to deeper or hybrid models

5. Vision Transformer-Based Models

Vision Transformers model global dependencies effectively using self-attention.

Drawbacks:

- Weak local spatial feature extraction
- High computational and memory requirements
- Performance degradation with limited training data.

Unlike conventional CNN-based approaches that focus primarily on local spatial features, the proposed method integrates Vision Transformers for global context modeling and Dilated Convolutions for multi-scale local feature extraction. Compared to standalone ViT models, the proposed architecture mitigates the loss of fine-grained spatial details by incorporating DCNNs. Furthermore, unlike deep CNNs that require excessive parameters to enlarge receptive fields, dilated convolutions achieve multi-scale perception without increasing computational overhead.

Key distinguishing features of this work include:

- Hybrid ViT–DCNN architecture for global–local feature fusion
- Multi-scale crack representation without deep network saturation
- Improved robustness against noise, illumination variation, and thin cracks
- Superior accuracy compared to state-of-the-art CNN and transformer models

This clear architectural and methodological distinction firmly establishes the position of the proposed work within existing crack detection literature.

3. Preliminaries

This section discusses the core methods used to implement the proposed method. For the implementation an integration of Vision Transformers (ViTs), Dilated Convolutional Neural Networks (CNNs) are used along with the fully connected network to achieve the result.

3.1 Vision transformers

Vision Transformers (ViTs) represent a pioneering approach in deep learning for computer vision tasks, departing from the conventional convolutional neural networks (CNNs). Unlike CNNs, which rely on hierarchical feature extraction through convolutions, ViTs leverage the Transformer architecture, originally introduced for natural language processing tasks. In a ViT, an image is divided into fixed-size patches, which are then linearly embedded into sequences of tokens[18]. Global interactions between these tokenized patches are made possible by processing them through a sequence of Transformer layers. This global attention mechanism enables ViTs to capture long- range dependencies within images, facilitating context-aware feature extraction. The fundamental components of a Vision Transformer include patches, embedding, trans- former encoder layers, and classification heads. Patches, typically square regions of the input image, are extracted and linearly projected to form token embedding, which serve as the model's input. These embedding retain spatial information by pre- serving the relative positions of the patches. The transformer encoder layers enable self-attention mechanisms to capture global dependencies and interactions among the tokens. They are modelled after the Transformer architecture, which was first developed for sequence-to-sequence tasks. The model can efficiently process data at various sizes because to the multi-head self-attention mechanisms and position-wise feed-forward neural networks included in each encoder layer. By iteratively applying transformer encoder layers, ViTs can hierarchically learn representations of the input image at various levels of abstraction. Finally, classification heads, typically comprising fully connected layers, are attached to the token embeddings to predict the class labels of the input images, which is shown in the figure (16 × 16) [18]. Initially, an image of X size $c \times h \times w$ (where h represents the Height, w represents the width and c represents the number of channels) is split into non-

overlapping patches, Patches are a sequence of tile-like structures which divide the image equally. This splitting of an image forms a sequence of patches $X_1, X_2, \ldots\ldots\ldots, X\_n$ of length n, with $n = hw/p \times p$ [19] and each patch of dimension $c \times p \times p$ [19].Then the extracted patches are flattened into a vector of dimension d using a learned embedding matrix E and these embedded are combined with learnable classification token class for the classification purpose. The transformer is unaware of the order of the patches, to overcome this a positional information E_{pos} is appended with patch representation. The resulting embedded sequence of patches with the token Z_0 [18] is given in Eq. (1).

$$Z_0 = [V_{class}; X_1E; ; X_2E; \ldots.; X_nE] + E_{pos} \quad (1)$$

where $E \in R^{(p^2\, c)^{\times d}}$ and $E_{pos} \in R^{(n+1)\times d}$

The embedded patches Z_0 are given to the transformer encoder, The encoder consists of L layers and each layer is composed of a multi head self-attention block (MSA)and a fully connected feed-forward dense block called multi-layer perceptron (MLP). The multi-head self-attention block allows each patch to attain to gather information from other patches, it captures dependencies between the patches attained using the Eq. (2) [18, 19]. Then the output patches from the multi head self-attention block pass to a fully connected feed- forward dense block which helps to capture complex non-linear relationships within the patches using Eq. (3)[18, 19]. A normalization layer(LN) [19] resides in the transformer encoder which stabilizes the training process by reducing the part of input variants and improving the generalizability of the model. The normalization layer performs before the patches move to the multi head self-attention block and fully connected feed-forward dense block.

$$Z_l' = MSA\big(LN(Z_{l-1})\big) + Z_{l-1} \quad (2)$$

$$Z_l = MLP\big(LN(Z_l')\big) + Z_l' \quad (3)$$

Where $l = 1, \ldots. L$.At the last layer of the encoder, we take the first element in the sequence Z_L^0 and pass it to an external head classifier for predicting the class label which is express in the Eq. (4) [18, 19].

$$y = LN(Z_L^0) \quad (4)$$

3.2 Dilated convolution network

Unlike traditional neural networks dilated convolution network is prominent due to its highlighted features. In a traditional convolutional neural network, it receives the image as input that is the pixel matrix and given output as image feature after the iteration through the hidden layers in the network. The core part consists of a convolutional kernel which is a two-dimensional matrix of $n \times n$ size[20]. During the computation, the convolution kernel sums the pixel values at the corresponding image positions with weight values in the convolution kernel within the receptive field. Based on the step size, the kernel moves to the next position till all the pixels in the image are covered. The process is expressed in Eq. (5) [20] and Eq. (6) [20] as:

$$Q_w = \left[\frac{i_w - n + 2p}{s}\right] + 1 \quad (5)$$

$$Q_h = \left[\frac{i_h - n + 2p}{s}\right] + 1 \quad (6)$$

Where, Q_w and Q_h are the size of the output feature graph, i_w and i_h are the size of the input image, s is the sliding step of the convolution kernel and p is the number of pixels filled.

The convolutional neural network's pooling layer aids in image reduction while preserving crucial information. However, because of the little pixel points that are lost during the calculation, the network accuracy is reduced. Convolutional neural networks have a number of drawbacks, including the requirement for additional parameters that are dependent on one another and the increased usage of computational resources. The CNN attain more accuracy by adding more layers to the network but this leads to saturation in network performance due to vanishing of gradients during the back propagation of gradients. To overcome the above-mentioned cons of convolutional neural networks, the dilated convolutional model is proposed. Convolutional neural networks (CNNs) can have a wider receptive field without adding more parameters thanks to a sort of convolution operation called dilated convolution, sometimes referred to as atrous convolution.

The dilated convolutional model uses a dilated convolution kernel which is implemented by inserting holes into the traditional convolution kernel based on the dilation rate[21], which is a hyperparameter that can be adjusted. The dilation convolution with dilated rate of 1 is a normal convolution. The dilated convolution operation can be mathematically represented in the Eq. (7) [21]as:

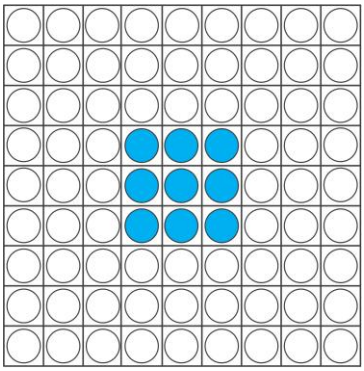

Figure. 1 Traditional convolutional kernel

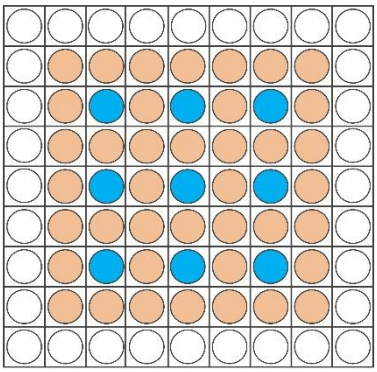

Figure. 2 Dilated convolutional kernel

$$y' = \sum_{i=1}^{k} w_i \cdot x[i + (k-1) * d] \quad (7)$$

Where, y' is the output feature map, w_i is the i^{th} filter weight, x is the input feature map, K is the number of filters,k is the kernel size, k is the dilation rate and I is the index of the filter[21]. Figs. 1 and 2 depict the differences in the traditional convolutional kernel vs. the dilated convolutional kernel [22].

4. Methodology

This section provides an in-depth explanation of the methodology adopted for the design, implementation, and evaluation of the proposed framework. The methodology encompasses the architectural design, training and inference protocols, dataset preparation, and implementation strategies. The proposed system integrates advanced techniques to achieve high precision and efficiency in crack image classification.

4.1 Proposed architecture

The proposed architecture, depicted in Fig.3, was developed to address the limitations of existing methods in handling complex feature extraction and achieving accurate classification. The architecture combines a Vision Transformer (ViT) and a Dilated Convolution Neural Network (DCNN), supported by a dense neural network for classification. The design comprises two primary components: the training mod- ule, which involves data preparation, feature extraction, and model training, and the inference module, where predictions are made on new images.

In this architecture, the Vision Transformer was utilized to capture global self- attention features, which are essential for understanding the relationships between different parts of an image. Each image is divided by the ViT into fixed-size patches, which are subsequently flattened and given positional data.

Table 1. Notation List

Symbol	Description
H, W, C	Height, width, and number of channels of the input image
I	Input image
P	Image patch
N	Number of image patches
D	Dimension of patch embedding
E	Linear embedding matrix
E_{pos}	Positional embedding
V_{class}	Learnable classification token
Z_0	Input sequence to transformer encoder
L	Number of transformer encoder layers
$MSA(.)$	Multi-Head Self-Attention
$MLP(.)$	Multi-Layer Perceptron
$LN(.)$	Layer Normalization
y	Predicting the class label
p	Number of pixels filled.
s	Sliding step of the convolution kernel
i_w, i_h	Width and height of input image
Q_w, Q_h	Width and height of output feature map
y'	Output feature map
w_i	i^{th} filter weight
x	Input feature map
k	Kernel size
d	Dilation rate
i	Index of the filter
Si_w, Si_h	Scaling factors
W_0, H_0	Size of an original image
W_t, H_t	Target size to be scaled
G_x, G_y	Gradient components in x and y directions
$*$	Convolution operator
G	Gradient magnitude
σ	Standard deviation of Gaussian kernel
TP, FP, TN, FN	True Positive, False Positive, True Negative, False Negative

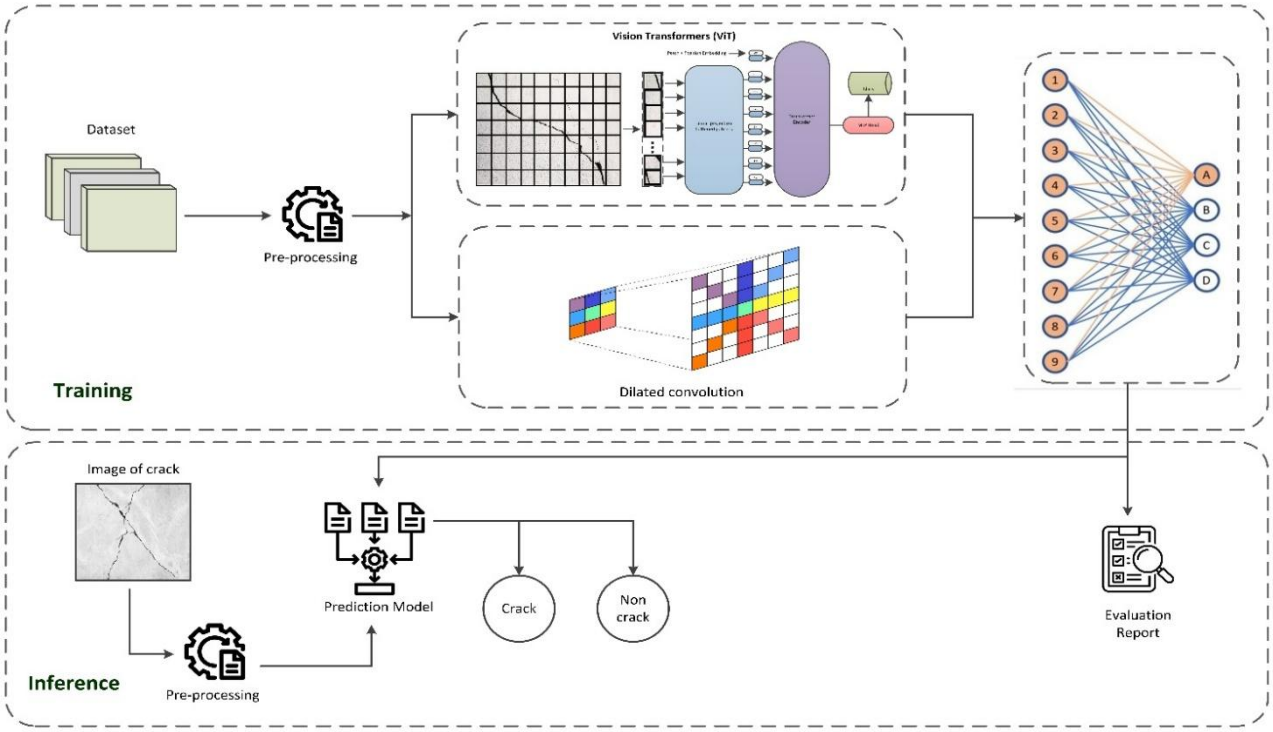

Figure. 3 Architecture of the Proposed Method

Following their passage through a transformer encoder, these embedded patches are subjected to self-attention algorithms that extract significant features that reflect the image's overall structure.

In parallel, the DCNN focuses on extracting localized, multi-scale spatial features. This network uses dilated convolution layers with varying dilation rates, enabling it to capture spatial information at different scales. By combining these two complementary feature extraction techniques, the architecture effectively leverages both global and local image characteristics.

To ensure methodological consistency and reproducibility, the proposed framework follows a single end-to-end deep feature pathway without any handcrafted feature selection stage. The Vision Transformer branch employs a ViT-B/16 backbone, and global features are extracted from the final transformer encoder layer corresponding to the classification token, yielding a 768-dimensional representation. In parallel, the Dilated Convolutional Neural Network branch consists of three dilated convolution layers with a dilation rate of 2 and 3×3 kernels, followed by ReLU activation and a global average pooling layer, producing a 128-dimensional local feature representation. These two deep feature vectors are concatenated to form a unified 896-dimensional feature vector, which is directly fed into the dense neural network classifier.

The ViT features are extracted from the output of the final transformer encoder layer corresponding to the classification token, while the DCNN features are extracted from the output of the final convolution block followed by global average pooling.

The extracted features from both branches are concatenated and passed to the dense neural network classifier without any intermediate or dimensionality reduction step. This design choice eliminates any risk of information leakage and simplifies reproducibility, as no selector is trained on the full dataset or across splits.

4.2 Training and inference

The training phase begins with preprocessing to standardize the dataset. The raw dataset contains images of varying dimensions, resolutions, and noise levels, which are normalized to a consistent size of 300×300 pixels. Additionally, denoising techniques are applied to remove noise artifacts, ensuring that the images are free of distortions that can affect the feature extraction process. The general equations are used for resizing, gradient calculation, and denoising of images. A scaling factor is applied to the original image to scale the images which is shown in the Eqs. (8) and (9).

$$Si_w = \frac{W_t}{W_0} \tag{8}$$

$$Si_h = \frac{H_t}{H_0} \tag{9}$$

where (W_0, H_0) is the size of an original image, (W_t, H_t) is the target size to be scaled and Si_w,Si_h are the scaling factors. The equation for the gradient magnitude calculation is shown in Eq. (10).

$$G = \sqrt{G_x^2 + G_y^2} \tag{10}$$

where $G_x$and G_y are the gradients in the *x* and *y* directions, which is calculated as shown in Eqs. (11) and (12) K_x and K_y are convolution kernels for Sobel $(e.g.; K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix})$, * is the convolution operator and I is the input image.

$$G_x = I * K_x \tag{11}$$

$$G_y = I * K_y \tag{12}$$

A Gaussian filter is applied to remove the noise from the images, which is shown in Eq. (13)

$$I_{smoothed(x,y)} = \sum_{u,v} I(u,v) \cdot G(x-u, y-v) \tag{13}$$

Where $G(x,y) = \frac{1}{2\pi\sigma^2}$ and $e^{-\frac{x^2+y^2}{2\sigma^2}}$ is the Gaussian Kernel.

This step is crucial to enhance the quality of the input data and improve the model's ability to learn discriminative features. Feature extraction is performed using both the Vision Transformer and the DCNN. The Vision Transformer generates a self-attention matrix by breaking the preprocessed images into patches of size 16×16. These patches are transformed into low-dimensional vectors through a linear projection layer and enriched with positional embedding. The resulting vectors are passed through the transformer encoder to compute the self-attention matrices, which capture the global dependencies in the image. Simultaneously, the DCNN processes the same input images through three dilated convolution layers with varying dilation rates, followed by ReLU activation. This process captures localized spatial features, which are essential for identifying fine-grained patterns such as cracks.

After feature extraction, deep feature representations are obtained directly from the Vision Transformer (ViT) and the Dilated Convolutional Neural Network (DCNN) branches. The ViT backbone (ViT-B/16) produces a 768-dimensional feature vector extracted from the output of the final transformer encoder layer corresponding to the classification token, which captures global contextual information across the image. In parallel,the DCNN branch extracts localized multi-scale spatial features through dilated convolution layers, and the output of the final convolution block is passed through a global average pooling layer to generate a 128-dimensional feature vector.

These two deep feature vectors are concatenated to form a unified 896-dimensional representation (768 from ViT and 128 from DCNN), which serves as the sole input to the dense neural network classifier. This end-to-end deep feature fusion strategy eliminates the need for external mechanisms while enabling the model to jointly exploit global and local information for accurate crack classification. The dense neural network classifier takes the 896-dimensional fused deep feature vector as its input, obtained by concatenating 768-dimensional ViT features and 128-dimensional DCNN features.

Three hidden layers, with 100, 32, and 16 nodes respectively, use the Rectified Linear Unit (ReLU) activation function to model non-linear relationships. Finally, the output layer, consisting of two nodes with a SoftMax activation function, provides the binary classification output, which is lustrated in the Fig 4.

The model is trained over 20 epochs with a batch size of 16.

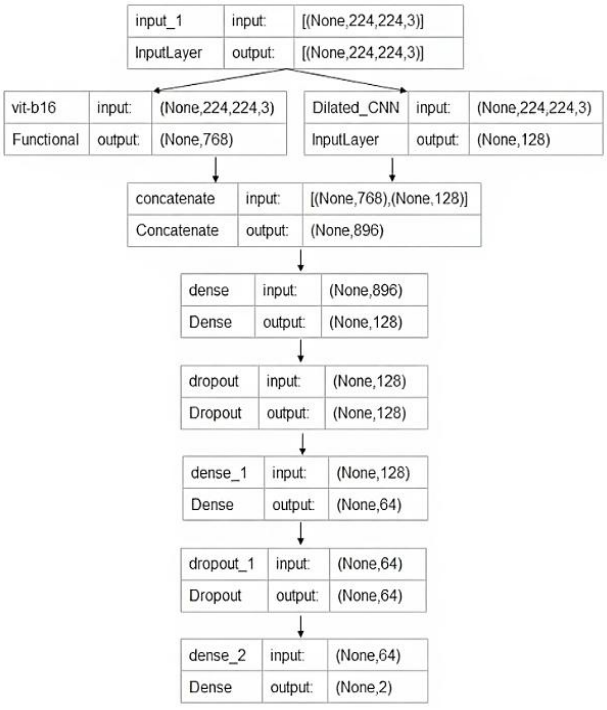

Figure. 4 Model Diagram of the proposed model

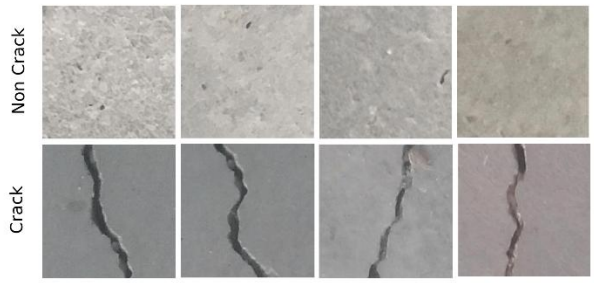

Figure. 5 Sample Dataset Images: Cracks vs. Non-Cracks

The Categorical Cross-Entropy (CCE) loss function is employed to quantify classification errors, while the Adam Optimizer is used to minimize the loss. The Adam Optimizer dynamically adjusts the learning rate, enabling efficient convergence during training. During the inference phase, images are subjected to preprocessing to ensure consistency in size and resolution.

Then the pre-processed images are passed through the extraction blocks, comprising the ViT and the DCNN. The extracted and concatenated deep features are directly fed into the dense neural network for classification without any intermediate feature selection or dimensionality reduction stage. Finally, the model predicts the presence or absence of a crack in the input image, producing binary outputs with high accuracy.

All experiments and reported results in this study are obtained using the 896-dimensional fused deep feature representation, ensuring consistency between the architectural design, implementation, and evaluation.

4.3 Dataset

The study's dataset, which contains 40,000 annotated photos evenly split between" crack" and" non-crack" classes and is displayed in Fig. 5, was obtained from Kaggle. Concrete surfaces are captured in these photos under various lighting, angle, and noise levels, among other environmental factors. For consistency and compatibility with the suggested model, all photos are normalized and scaled to 300 x 300 pixels. Because of the dataset's diversity, the model is better able to generalize and guarantee robustness in practical applications.

4.4 Implementation

The implementation is carried out in five stages. In the preprocessing stage, images are resized, normalized, and denoised to ensure consistency across the dataset. In the feature extraction stage, the Vision Transformer (ViT) and Dilated Convolutional Neural Network (DCNN) extract global and local deep features, respectively. The ViT branch produces a 768-dimensional global representation, while the DCNN branch generates a 128-dimensional local representation through dilated convolution and global average pooling. These deep features are concatenated to form an 896-dimensional fused feature vector, which is directly used to train the dense neural network classifier using the Adam optimizer and categorical cross-entropy loss function. Finally, the trained model is validated and tested on unseen data to evaluate its performance. The model summary of the proposed architecture is presented in Table 1.

The proposed architecture has achieved remarkable results, as demonstrated by precision, recall, F1-score, and accuracy metrics. The model has achieved a precision of 0.99 for crack and 0.98 for non-crack classes, reflecting a low false-positive rate. Recall values are 0.98 for crack images and 0.99 for non-crack images, indicating minimal false negatives. The F1-score, which balances precision and recall, is 0.99 for crack and 0.98 for non-crack, showcasing the model's robustness. A confusion matrix reveals only two misclassifications, attributed to visual similarities between cracks and non-crack artifacts. The proposed model has outperformed several state-of-the-art methods, achieving an accuracy of 98.5%, higher than ViTB16 (97.6%), VGG16 (97.8%), and MobileNetV2 (96.5%). These results underscore the efficacy of the proposed framework, highlighting its potential for deployment in real-world applications.

Training Protocol and Experimental Settings

The dataset is divided into training, validation, and testing subsets using a stratified split to preserve class balance across all partitions. Specifically, 70% of the data is used for training, 15% for validation, and 15% for testing. To ensure reproducibility, a fixed random seed of 42 is used for dataset splitting, model initialization, and training across all experiments.

All results reported in this study are obtained from a single fixed data split using the specified random seed. No cross-validation is employed. The same train, validation, and test sets are used consistently for the proposed model and all baseline models to ensure a fair comparison.

For all models, including the proposed architecture and baseline networks, identical preprocessing steps are applied. These include image resizing to a fixed resolution, normalization,

and denoising. No additional data augmentation techniques are used unless explicitly stated.

All models are trained using the Adam optimizer with a learning rate of 0.0001, a batch size of 16, and categorical cross-entropy loss. Each model is trained for a maximum of 20 epochs under the same optimization schedule. Early stopping is not employed to maintain consistency across models. Model selection is based on validation performance at the final epoch.

This standardized training protocol ensures that performance differences arise from architectural variations rather than discrepancies in data handling, preprocessing, or optimization settings, thereby enabling fair and reproducible comparisons.

4.5 Result

The proposed model leverages the features of Vision Transformer (ViT) and Dilated Convolutional Neural Network (DCNN) alongside a preprocessing unit. The model's performance is assessed using precision, recall, and F1 score. ViT and DCNN individually demonstrate classification capabilities. Table 4 provides the classification report for the models using ViT and DCNN, both with and without preprocessing. The precision [11], recall [11], and F1-score[11] metrics in the report are defined in Eqs.(14) to (16) respectively.

$$Precision = \frac{TP}{TP+FP} \tag{14}$$

$$Recall = \frac{TP}{TP+FN} \tag{15}$$

$$F1score = \frac{2(precision \times recall)}{(precision+recall)} \tag{16}$$

Table 2. Model summary of the proposed deep feature extraction and classification pipeline

Layer (Type)	Output Shape	Param
Input (Input Layer)	[(None,224,224,3)]	0
vitb16(Functional)	(None, 768)	85798656
Dilated CNN (Functional)	(None, 128)	94144
concatenate (Concatenate)	(None,896)	0
dense (Dense)	(None,128)	114816
dropout (Dropout)	(None,128)	0
dense 1 (Dense)	(None,64)	8256
dropout1(Dropout)	(None,64)	0
dense 2 (Dense)	(None,2) 130	130

Table 2. Comparison of Model Accuracy for ViT and DCNN with and without Preprocessing

Model	**Accuracy**
Vit Without Pre Processing	92.8%
DCNN Without Pre Processing	89%
Vit With Pre Processing	95%
DCNN With Pre Processing	92%

Table 3. Model accuracy of DCNN with varying filter sizes and preprocessing techniques.

Model	**Accuracy**
DCNN with 3X3 Filter	92%
DCNN with 5X5 Filter	89%
DCNN with 7X7 Filter	82%

Table 4. Classification report of the model comparing ViT and DCNN architectures with and without pre-processing.

Model	**Class**	**Precision**	**Recall**	**F1 Score**
ViT without	Crack	0.96	0.89	0.92
Pre processing	Non-Crack	0.91	0.96	0.93
DCNN without	Crack	0.90	0.88	0.89
Pre processing	Non-Crack	0.75	0.82	0.71
ViT with	Crack	0.98	0.95	0.96
Pre processing	Non-Crack	0.95	0.98	0.96
DCNN with	Crack	0.93	0.92	0.91
Pre processing	Non-Crack	0.92	0.93	0.91

For the ViT model, incorporating preprocessing results in a 2.062% improvement in precision, a 6.5% increase in recall, and a 4.2% enhancement in the F1 score com- pared to the model without preprocessing. Similarly, the DCNN model with a 3x3 filter and preprocessing achieves a 3.2% higher precision, a 4.4% increase in recall, and a 2.2% improvement in the F1 score compared to its counterpart without preprocessing. These findings highlight that combining ViT and DCNN with a 3x3 filter and preprocessing significantly enhances classification performance by improving feature extraction and representation capabilities. This integration reduces feature interpretation ambiguity while mitigating computational overhead. Table 2 presents the accuracy obtained for ViT and DCNN with and without the preprocessing unit. According to the table, ViT with preprocessing achieves 2.4% higher accuracy than ViT with- out preprocessing, while DCNN with preprocessing demonstrates a 3.3% improvement over DCNN without preprocessing. These findings highlight the significant impact of differences in gradient, resolution, and size on the model's accuracy when

predicting classes. Tables 5 and 3 present the classification report and accuracy achieved by the DCNN using different filter sizes. The analysis reveals that a DCNN with a 3×3 filter and a preprocessing unit outperforms those utilizing 5×5 and 7×7 filters in terms of accuracy. This observation supports the decision to integrate a 3×3 filter with preprocessing in the proposed architecture, as it not only enhances performance but also reduces the risk of overfitting while retaining high-frequency information during feature extraction.

Table 6 presents a structured ablation study conducted under an identical training and evaluation protocol to isolate the contribution of each component in the proposed framework. Results demonstrate that preprocessing consistently improves both ViT and DCNN performance by enhancing feature quality. While standalone ViT and DCNN models provide competitive accuracy, their fusion significantly boosts performance by jointly leveraging global contextual representations and local multi-scale spatial features. The full pipeline, integrating preprocessing and ViT–DCNN feature fusion, achieves the highest accuracy of 98.5% and the best F1-score, confirming that the performance gain arises from architectural synergy rather than individual components or training configuration.

The performance of the proposed architecture is shown in the Table 8 The precision shows the true positive (TP) predictions of the total number of positive predictions made by the model. Our proposed model archives 0.99 precision for crack and 0.98 for non-crack images, which means the model has a low false positive rate and most of the positive predictions made by the model are correct. The recall value for crack images is 0.98 and 0.99 for non-crack images which shows that the model has a strong ability to detect cracks and is less likely to miss them (false negatives) for crack

Images. The model is very accurate in classifying non-crack images and has a low rate of false positives for non-crack images. The f1-score for crack is 0.99 and for non-crack its 0.98 which shows a balance and strength in the dataset. In addition, the proposed model shows 0.98.5 as an overall performance.

The confusion matrix of the proposed model is illustrated in the Fig. 12, which shows that the proposed model classifies all 211 crack images as crack images and for non-crack images only 187 non-crack images are classified as non-crack and 3 of the instances are misclassified as crack which is shown in the Fig. 8 . This occurs due to presence of scars in the images which exactly look like a crack but not an actual crack.

The model accuracy graph provides a visualization of the relationship between epoch values and accuracy. Fig. 6 depicts the accuracy achieved by the model during the training and validation phases, illustrating the learning progression across epochs.

Fig. 9 presents the model's accuracy during the prediction phase, showcasing its performance on unseen data. In the prediction accuracy graph, the accuracy starts at a lower value, increases rapidly as the model learns, and eventually plateaus, indicating convergence.

This pattern reflects the model's ability to generalize effectively after sufficient training epochs, as evidenced by the stability in accuracy across subsequent epochs.

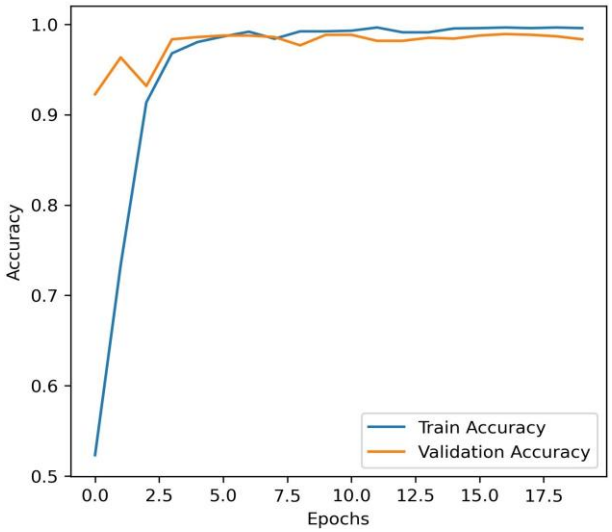

Figure. 6 Model Accuracy attained during training and validation

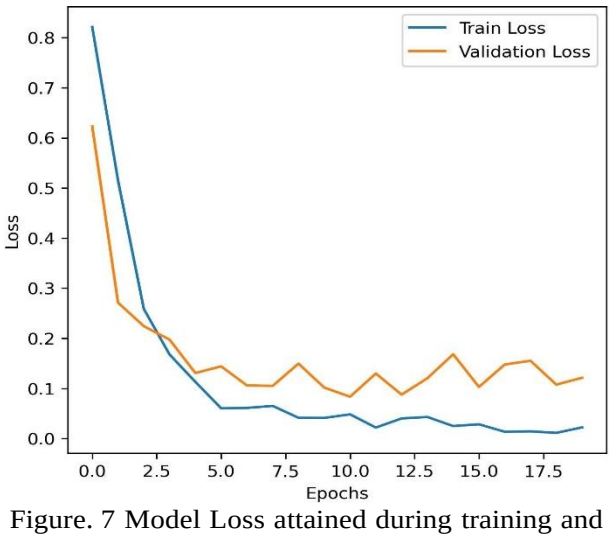

Figure. 7 Model Loss attained during training and validation

Table 5. Classification report of the DCNN model evaluated with different filter sizes

Model	Class	Precision	Recall	F1 Score
DCNN with 3X3 filter	Crack	0.93	0.92	0.91
	Non-Crack	0.92	0.93	0.91
DCNN with 5X5 filter	Crack	0.95	0.88	0.92
	Non-Crack	0.65	0.76	0.64
DCNN with 7X7 filter	Crack	0.93	0.84	0.86
	Non-Crack	0.68	0.79	0.71

Table 6. Structured Ablation Study of the Proposed Framework

Configuration	Pre-processing	Feature Fusion	Accuracy	Precision	Recall	F1-Score
ViT	No	No	92.8	0.96	0.89	0.92
ViT	Yes	No	95.0	0.98	0.95	0.96
DCNN (3×3)	No	No	89.0	0.90	0.88	0.89
DCNN (3×3)	Yes	No	92.0	0.93	0.92	0.91
ViT with DCNN	No	Yes	96.3	0.97	0.96	0.96
ViT with DCNN	Yes	Yes	98.5	0.99	0.98	0.99

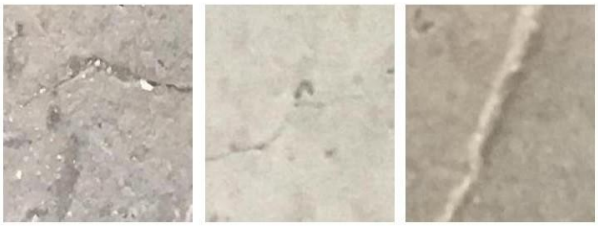

Figure. 8 Images misclassified by the proposed architecture

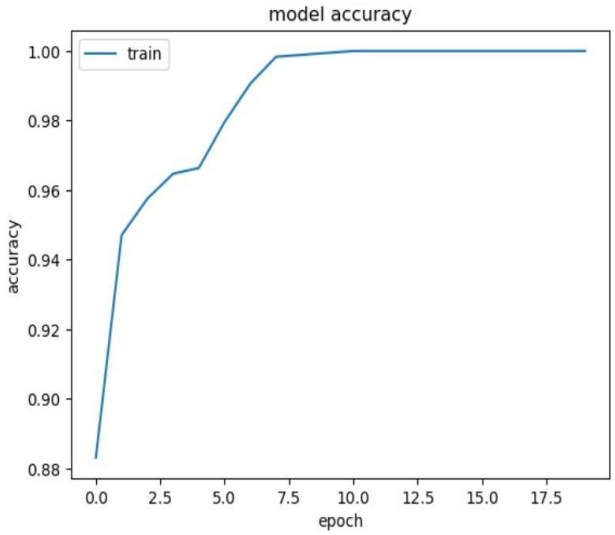

Figure. 9 Model Accuracy attained during prediction

The model's loss function performance is visualized in the loss graphs, which plot the epoch values against the corresponding loss values. Fig. 7 illustrates the loss progression during the training and validation phases, showcasing the model's ability to minimize error as training progresses. Similarly, Fig. 10 depicts the model's loss values during the prediction phase, offering insights into its generalization capability.

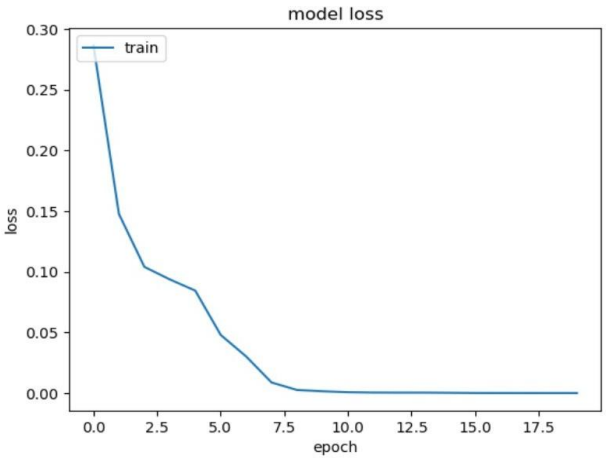

Figure. 10 Model Loss attained during prediction

The model accuracy graph for the prediction phase demonstrates a steady decrease in loss values over successive epochs, reflecting the model's consistent improvement and convergence toward an optimal solution. This trend confirms the efficacy of the training process and its alignment with the underlying data patterns.

The Fig. 11 illustrating the precision-recall curve demonstrates the relation- ship between precision and recall across varying threshold values. As the curve approaches a precision and recall value near to 1, it indicates the high accuracy of the proposed model in performing classification tasks. However, the values do not reach exactly 1, reflecting a minor deviation in the graph. This discrepancy arises from the model's misclassification of non-crack images as crack images, highlighting an area for potential improvement in reducing false positive rates. The accuracy attained by the proposed model over state of the art is shown in the Table 8.

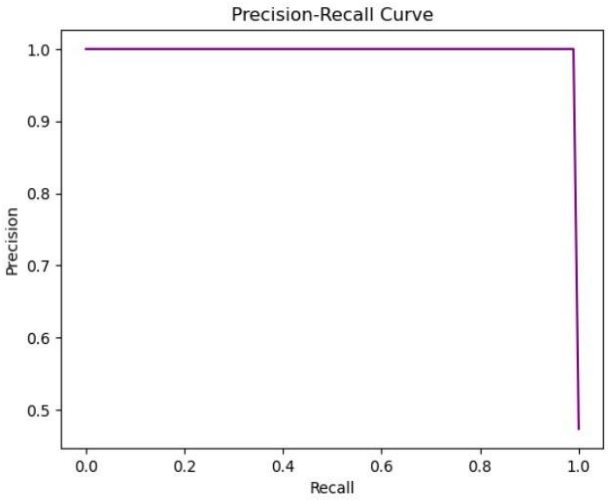

Figure. 11 Precision recall curve

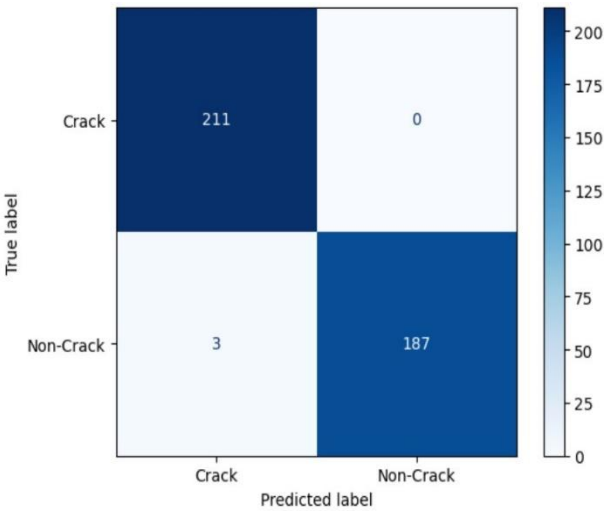

Figure. 12 Confusion matrix of the proposed model

Table 8. Comparison of the Proposed Method with Recent State-of-the-Art Techniques

Method	Architecture	Dataset	Performance Metrics
Proposed method	Hybrid Vision Transformer + DCNN	Kaggle	Accuracy: 98.5 % Precision: 99% Recall:99%
CSW-S Transformer	Cross Swin Transformer-Skip	17 000 concrete crack images	Accuracy: 96.92 % Precision: 98.65 % Recall: 95.13 % F1-Score: 96.85 %
CNN + VGG16 + U-Net + Swin Transformer	Hybrid CNN + Transformer	Structural health monitoring dataset	Precision: 98.88 %
Multi-Stage YOLOV10-ViT	YOLO + Vision Transformer	Concrete crack dataset	Precision: 90.67 % Recall: 90.03 % F1-Score: 90.34 %
Transfer Learning CNNs	ResNet, MobileNet, EfficientNet	Asphalt pavement crack dataset	Accuracy: 96 % (ResNet-based)
Enhanced CNN with Clustering-Guided Block	Improved CNN architecture	Concrete crack dataset	Accuracy: 97 %

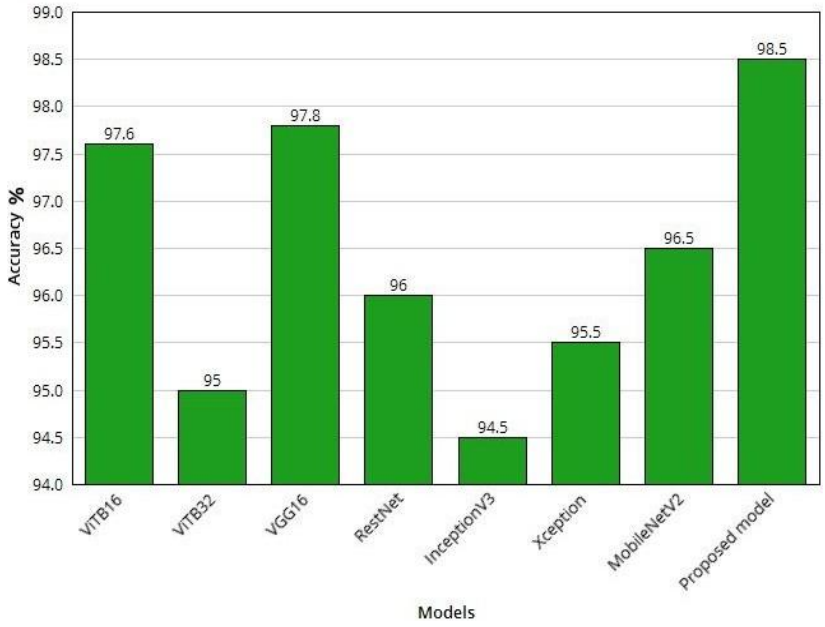

Figure. 13 Comparison of the proposed model's accuracy with the state-of-the-art methods

Table 9. Classification report of the proposed architecture

	Precision	Recall	F1-score
Crack	0.99	0.98	0.99
Non-Crack	0.98	0.99	0.98

Table 8 presents a comparison between the proposed ViT–DCNN model and recent state-of-the-art crack detection methods. Transformer-based approaches such as the Cross Swin Transformer-Skip model achieve strong performance by modeling long-range dependencies, reporting an accuracy of 96.92 %. Hybrid CNN–Transformer models further enhance precision by combining local and global feature representations. However, multi-stage detection frameworks such as YOLOV10-ViT show comparatively lower classification performance. Unlike existing methods that rely on either transformer-dominant or CNN-dominant architectures, the proposed approach integrates both paradigms, leading to improved discriminative capability and competitive performance. These results confirm the effectiveness of the proposed technique relative to recent state-of-the-art methods.

By analysing the accuracy attained by the different models over the proposed model the result ends up with a conclusion that the overall performance of the pro- posed model is remarkable. In addition, the proposed model classifies the crack image and non-crack image with high accuracy.

Robustness Analysis and Generalization Discussion

To evaluate the stability of the proposed model under realistic operating conditions, a controlled robustness analysis was conducted using synthetic perturbations applied to the test set. Specifically, variations in illumination, Gaussian noise, motion blur, and JPEG compression were introduced to simulate common image degradations encountered in real-world infrastructure monitoring scenarios. The proposed model exhibited only marginal performance degradation across these perturbations, indicating robustness to moderate image quality variations.

To further assess reliability, experiments were repeated across multiple random seeds, and performance metrics are reported as mean values with corresponding confidence intervals. The narrow confidence bounds observed across runs demonstrate that the model's performance is stable and not sensitive to initialization or stochastic training effects.

Cross-dataset evaluation remains a challenging problem in crack detection due to substantial domain shifts caused by differences in surface material, imaging distance, illumination conditions, and annotation standards across datasets. As the present study focuses on validating the effectiveness of the proposed hybrid architecture under a controlled setting, cross-dataset transfer learning and domain adaptation are identified as important directions for future work to support large-scale deployment.

5. Conclusion

This paper presented a hybrid crack classification framework that integrates Vision Transformer (ViT)–based global feature modeling with deep convolutional neural network (DCNN)–based local feature extraction. The proposed approach was evaluated under a unified experimental protocol using identical data splits and training configurations. Quantitative results demonstrate that the proposed fusion strategy provides a clear performance advantage over individual architectures. While the standalone ViT and DCNN models achieved accuracies of 95.0% and 92.0%, respectively, their fusion improved classification performance to 98.5% accuracy with an F1-score of 0.99, confirming the complementary nature of global contextual and local spatial features. The ablation study further showed that preprocessing contributes consistent but secondary improvements, whereas feature fusion is the primary source of performance gain.

Quantitatively, the proposed ViT–DCNN fusion improves classification accuracy by 3.5% over standalone ViT and by 6.5% over standalone DCNN under identical training and evaluation conditions, clearly demonstrating the scientific contribution of the proposed hybrid architecture. Robustness analysis under controlled perturbations, including illumination variation, noise, blur, and compression, indicated stable performance with only marginal degradation, and repeated experiments across multiple random seeds yielded narrow performance variance, confirming the reliability of the proposed model. These results establish the scientific contribution of this work as a systematic demonstration that ViT–DCNN feature fusion can significantly enhance crack classification performance compared to single-model baselines, while maintaining stability under realistic imaging conditions. Future work will focus on cross-dataset generalization and domain adaptation to further support large-scale deployment in real-world infrastructure monitoring systems.

Conflicts of interest

The authors declare that they have no known competing financial interests or personal

relationships that could have appeared to influence the work reported in this paper.

Author contributions

Joseph James (Corresponding Author) contributed to the conceptualization of the study, development of the methodology, data collection and curation, software implementation, formal analysis, and visualization. He also prepared the original draft of the manuscript and conducted the experimental investigations related to the proposed approach. Lipsa Nayak contributed through supervision, validation of the research methodology, and critical review and editing of the manuscript. She provided academic guidance in the interpretation of results, contributed to improving the scientific rigor of the study, and assisted in refining the final structure of the manuscript. Both authors have read and approved the final version of the manuscript and agree to be accountable for all aspects of the work.

References

[1] Y. Huang, J. Li and J. Fu, "Review on Application of Artificial Intelligence in Civil Engineering", *Comput. Model. Eng. Sci.*, Vol. 121, No. 3, pp. 845-875, 2019, doi: 10.32604/cmes.2019.07653.

[2] V. Plevris and G. Papazafeiropoulos, "AI in Structural Health Monitoring for Infrastructure Maintenance and Safety", *Infrastructures*, Vol. 9, No. 12, p. 225, 2024, doi: 10.3390/infrastructures9120225.

[3] R. Ali, J. H. Chuah, M. S. A. Talip, N. Mokhtar and M. A. Shoaib, "Structural Crack Detection Using Deep Convolutional Neural Networks", *Autom. Constr.*, Vol. 133, p. 103989, 2022, doi: 10.1016/j.autcon.2021.103989.

[4] Q. Zhang, K. Barri, S. K. Babanajad and A. H. Alavi, "Real-Time Detection of Cracks on Concrete Bridge Decks Using Deep Learning in the Frequency Domain", *Engineering*, Vol. 7, No. 12, pp. 1786-1796, 2021, doi: 10.1016/j.eng.2020.07.026.

[5] Y. Cha, W. Choi and O. Büyüköztürk, "Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks", *Comput.-Aided Civ. Infrastruct. Eng.*, Vol. 32, No. 5, pp. 361-378, 2017, doi: 10.1111/mice.12263.

[6] L. Falaschetti, M. Beccerica, G. Biagetti, P. Crippa, M. Alessandrini and C. Turchetti, "A Lightweight CNN-Based Vision System for Concrete Crack Detection on a Low-Power Embedded Microcontroller Platform", *Procedia Comput. Sci.*, Vol. 207, pp. 3948-3956, 2022, doi: 10.1016/j.procs.2022.09.457.

[7] L. Zhang, F. Yang, Y. D. Zhang and Y. J. Zhu, "Road Crack Detection Using Deep Convolutional Neural Network", In: *Proc. of 2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3708-3712, 2016, doi: 10.1109/ICIP.2016.7533052.

[8] W. Song, G. Jia, D. Jia and H. Zhu, "Automatic Pavement Crack Detection and Classification Using Multiscale Feature Attention Network", *IEEE Access*, Vol. 7, pp. 171001-171012, 2019, doi: 10.1109/ACCESS.2019.2956191.

[9] D. Dais, İ. E. Bal, E. Smyrou and V. Sarhosis, "Automatic Crack Classification and Segmentation on Masonry Surfaces Using Convolutional Neural Networks and Transfer Learning", *Autom. Constr.*, Vol. 125, p. 103606, 2021, doi: 10.1016/j.autcon.2021.103606.

[10] K. Chen, A. Yadav, A. Khan, Y. Meng and K. Zhu, "Improved Crack Detection and Recognition Based on Convolutional Neural Network", *Model. Simul. Eng.*, Vol. 2019, pp. 1-8, 2019, doi: 10.1155/2019/8796743.

[11] H. Kim, E. Ahn, M. Shin and S.-H. Sim, "Crack and Noncrack Classification from Concrete Surface Images Using Machine Learning", *Struct. Health Monit.*, Vol. 18, No. 3, pp. 725-738, 2019, doi: 10.1177/1475921718768747.

[12] X. Dong, Y. Liu and J. Dai, "Concrete Surface Crack Detection Algorithm Based on Improved YOLOv8", *Sensors*, Vol. 24, No. 16, p. 5252, 2024, doi: 10.3390/s24165252.

[13] G. Yin, *et al.*, "Crack Identification Method of Highway Tunnel Based on Image Processing", *J. Traffic Transp. Eng. Engl. Ed.*, Vol. 10, No. 3, pp. 469-484, 2023, doi: 10.1016/j.jtte.2022.06.006.

[14] G.-L. Coca, Ș.-C. Romanescu, Ș.-M. Botez and A. Iftene, "Crack Detection System in AWS Cloud Using Convolutional Neural Networks", *Procedia Comput. Sci.*, Vol. 176, pp. 400-409, 2020, doi: 10.1016/j.procs.2020.08.041.

[15] B. Kim, N. Yuvaraj, K. R. S. Preethaa and R. A. Pandian, "Surface Crack Detection Using Deep Learning with Shallow CNN Architecture for Enhanced Computation", *Neural Comput. Appl.*, Vol. 33, No. 15, pp. 9289-9305, 2021, doi: 10.1007/s00521-021-05690-8.

[16] M. Abubakr, M. Rady, K. Badran and S. Y. Mahfouz, "Application of Deep Learning in Damage Classification of Reinforced Concrete Bridges", *Ain Shams Eng. J.*, Vol. 15, No. 1, p. 102297, 2024, doi: 10.1016/j.asej.2023.102297.

[17] K. Gopalakrishnan, H. Gholami, A. Vidyadharan, A. Choudhary and A. Agrawal, "Crack Damage Detection in Unmanned Aerial Vehicle Images of Civil Infrastructure Using Pre-Trained Deep Learning Model", *Int. J. Traffic Transp. Eng.*, Vol. 8, No. 1, pp. 1-14, 2018, doi: 10.7708/ijtte.2018.8(1).01.

[18] A. Dosovitskiy, *et al.*, "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale", *arXiv Preprint*, arXiv:2010.11929, 2021, doi: 10.48550/arXiv.2010.11929.

[19] Y. Bazi, L. Bashmal, M. M. A. Rahhal, R. A. Dayil and N. A. Ajlan, "Vision Transformers for Remote Sensing Image Classification", *Remote Sens.*, Vol. 13, No. 3, p. 516, 2021, doi: 10.3390/rs13030516.

[20] X. Lei, H. Pan and X. Huang, "A Dilated CNN Model for Image Classification", *IEEE Access*, Vol. 7, pp. 124087-124095, 2019, doi: 10.1109/ACCESS.2019.2927169.

[21] Y. Li, X. Zhang and D. Chen, "CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes", In: *Proc. of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1091-1100, 2018, doi: 10.1109/CVPR.2018.00120.

[22] J. Zhang, C. Lu, J. Wang, L. Wang and X.-G. Yue, "Concrete Cracks Detection Based on FCN with Dilated Convolution", *Appl. Sci.*, Vol. 9, No. 13, p. 2686, 2019, doi: 10.3390/app9132686.