Engineering Journal issn: 0125-8281

*Article*

# Enhancing Infrastructure Safety: A UAV-Based Approach for Crack Detection

**Bandla Pavan Babu[1,*], Sarah Khandagale[2], Vedashree Shinde[2], Saurach Gargote[2], and Kishore Bingi[3]**

1 School of Computing Science and Engineering, VIT Bhopal University, Madhya Pradesh, India
2 School of Computer Science and Engineering, D Y Patil International University, Pune, Maharashtra, India
3 Department of Electrical and Electronics Engineering, Universiti Teknologi PETRONAS, Seri Iskandar, Malaysia
*E-mail: bandlapavanbabu@vitbhopal.ac.in (Corresponding author)

**Abstract.** The imperative task of identifying and promptly detecting cracks in concrete bridges is crucial for preserving their structural health and ensuring the safety of users. Traditional bridge inspection methods heavily rely on human eyes and additional tools, demanding extensive training for inspectors and resulting in time-consuming processes. The increasing demand for Unmanned Aerial Vehicles (UAVs) has provided a transformative solution to access hard-to-reach areas efficiently. This research explores the integration of deep learning algorithms, including CNN, RCNN, Fast RCNN, Faster RCNN, and YOLO, to enhance the accuracy and efficiency of UAV-based crack detection systems. Experimental results affirm the effectiveness of these algorithms in addressing challenges such as lighting variations and small crack detection. The study aims to contribute to structural health monitoring, improving maintenance practices, and enhancing safety.

**Keywords:** Unmanned aerial vehicle, bridge inspection, crack detection, deep learning, CNN, RCNN.

**ENGINEERING JOURNAL** Volume 27 Issue 12
Received 13 June 2023
Accepted 7 December 2023
Published 31 December 2023
Online at https://engj.org/
DOI:10.4186/ej.2023.27.12.11

DOI:10.4186/ej.2023.27.12.11

## 1. Introduction

India has experienced remarkable growth in large-scale infrastructure projects over the past two decades, encompassing the construction of apartment complexes, commercial buildings, and transportation networks such as highways, railways, and bridges. This rapid development has underscored the importance of regular maintenance and inspection to ensure the structural integrity of these structures and mitigate risks to public safety, particularly for bridges that are exposed to weather conditions and heavy loads. Thorough inspections are crucial in identifying signs of wear and tear that could lead to structural failure or collapse, ultimately safeguarding public safety.

Structural Health Monitoring (SHM) has emerged as a systematic and essential process that plays a pivotal role in ensuring the safety, reliability, and optimal performance of civil structures. By continuously monitoring, assessing, and analyzing various structural parameters, SHM provides valuable insights into the condition and behavior of structures over time. However, most existing crack detection systems heavily rely on human observation in the context of bridge inspections. These inspections demand extensive training for inspectors, consuming significant time and resources. Moreover, examining the lateral sides and underside of bridges often requires the use of specialized bridge inspection units, resulting in additional costs. Despite the utilization of these units, working at heights and in challenging environments remains risky. Furthermore, the subjective nature of manual inspections introduces a degree of subjectivity and lacks objectivity in quantitative analysis, heavily relying on the experience and expertise of the surveyor.

In recent years, Unmanned Aerial Vehicles (UAVs), commonly known as drones, have emerged as a transformative technology with profound implications across various industries. Drones offer an unparalleled aerial perspective, facilitating efficient data collection, monitoring, and analysis in diverse fields. Their increasing utilization is driven by their unique capabilities and versatile applications, revolutionizing sectors such as aerial photography, disaster management, agriculture, and infrastructure inspection. Drones provide cost-effective, time-efficient, and safer alternatives to conventional methods.

One key advantage of drones is their ability to access remote or hard-to-reach areas, eliminating the need for manual intervention and reducing human risks. This accessibility makes drones invaluable in search and rescue missions, environmental monitoring, and infrastructure inspections, particularly in challenging or hazardous locations. Leveraging their maneuverability and advanced navigation systems, drones can gather valuable insights and data from previously inaccessible vantage points, eliminating the need for extensive resources. Advancements in drone technology, including improvements in flight stability, battery life, payload capacity, and sensor capabilities, have further enhanced their utility. Equipped with high-resolution cameras, thermal imaging sensors, LiDAR systems, and other sophisticated tools, drones enable precise data collection, mapping, and analysis. This data can then be processed using advanced algorithms such as Convolutional Neural Networks (CNN) and Region-based Convolutional Neural Networks (RCNN) to accurately identify and analyze cracks in captured images or collected data.

The detection of cracks plays a vital role in the inspection and monitoring of civil structures, allowing engineers and inspectors to proactively address potential issues before they escalate. Cracks can result from material deterioration, structural stress, or environmental conditions, and detecting them early on is crucial to ensure structural integrity. The integration of UAVs with deep learning methods further enhances crack detection capabilities by training neural networks to automatically identify and classify cracks

This research paper aims to study different types of algorithms used for crack detection and enhance the accuracy and efficiency of crack detection by utilizing algorithms such as Convolutional Neural Networks (CNN) and Region-based Convolutional Neural Networks (RCNN). By harnessing the capabilities of drones and advanced algorithms, this study seeks to contribute to the field of structural health monitoring, improving maintenance practices, and enhancing safety.

To facilitate the implementation and training of Convolutional Neural Networks (CNN) and Region-based Convolutional Neural Networks (RCNN) for crack detection, this research paper utilizes Jupyter Notebook as a powerful and versatile tool. Jupyter Notebook provides an interactive computing environment that allows for the creation and execution of code cells, as well as the integration of explanatory text and visualizations. This platform supports various programming languages, including Python, which is widely used in the field of deep learning. By leveraging the power and versatility of Jupyter Notebook, this research paper conducts the execution and refinement of the crack detection algorithm like CNN and RCNN. The interactive computing environment provided by Jupyter Notebook enables seamless experimentation by allowing the adjustment of hyper parameters and analysis of results. Through this iterative process, the algorithm's performance is optimized, leading to improved accuracy and efficiency in detecting cracks.

## 2. Literature Review

Convolutional Neural Networks (CNNs) have revolutionized crack detection in bridges, offering high accuracy and the ability to extract meaningful features from images. In 2019, Li et al. proposed a CNN-based approach that achieved an impressive accuracy of 99.06% [3]. Their method employed deep convolutional layers to capture intricate crack patterns, enabling the detection of diverse crack types and orientations. This breakthrough

demonstrated the efficacy of CNNs in identifying cracks with high precision, providing valuable insights for bridge inspectors and maintenance teams.

In a similar vein, a study conducted in 2021 by researchers [2] reported a remarkable accuracy of 97.20% using a CNN-based approach. Their research focused on addressing one of the challenges in crack detection—lighting variations. By incorporating robust preprocessing techniques and data augmentation methods, the CNN model proved resilient to changes in lighting conditions, ensuring reliable crack detection across different environments.

Building on these advancements, Chen et al. (2021) introduced a CNN-based method that achieved an accuracy of 98.00% [1]. Their approach surpassed traditional crack detection methods by effectively identifying small cracks that often go unnoticed. By leveraging the CNN's ability to capture intricate details and spatial relationships, Chen et al. enhanced the sensitivity of crack detection, leading to improved bridge inspection accuracy and safety.

Moreover, in a most recent study by [21] A. Rahai (2022) implemented a deep convolutional neural network (DCNN) with transfer learning (TF) technique for crack detection. The study explored various deep learning networks, conducting experiments on test images that demonstrated an accuracy of over 99% in damage detection.

Moving on to Region-based Convolutional Neural Networks (RCNNs), they have gained prominence in crack detection due to their ability to identify crack regions and accurately classify them. In 2016, Dong et al. achieved an accuracy of 96.20% using an RCNN algorithm [6]. Their method incorporated multiple scales of image features, enabling the detection of cracks with varying widths and orientations. By considering the contextual information surrounding cracks, the multi-scale RCNN demonstrated robust performance across different types of cracks, contributing to more comprehensive structural health monitoring.

In 2021, Dong et al. further advanced RCNN-based crack detection by addressing challenges related to lighting variations, camera angles, and complex crack patterns [4]. Their RCNN algorithm achieved an accuracy of 95.78%, highlighting its robustness in challenging real-world scenarios. Through the utilization of advanced feature extraction techniques and effective crack region proposals, Dong et al. enhanced the accuracy and reliability of crack detection systems, reducing false positives and improving overall inspection efficiency.

In a different approach, Zhang et al. (2020) employed a multi-task RCNN model that reported an accuracy of 96% [5]. Their research focused on detecting cracks of various shapes, including narrow cracks and those with low contrast. By incorporating multiple subtasks such as crack localization and crack type classification, the multi-task RCNN demonstrated superior performance in accurately detecting different crack geometries and characteristics.

Fast RCNN algorithms have also shown remarkable performance in crack detection, offering a balance between accuracy and efficiency. In 2018, Yang et al. proposed a Fast RCNN algorithm that achieved an accuracy of 93.40% [7]. Their work primarily focused on detecting small cracks and cracks with low contrast, which pose challenges for traditional detection methods. By leveraging the region proposal network and fast feature extraction, the Fast RCNN algorithm demonstrated its effectiveness in capturing fine-grained crack details, improving the overall accuracy of bridge inspection systems.

Li et al. (2022) further improved the accuracy of Fast RCNN-based crack detection by addressing cracks with various orientations and shapes, including diagonal and corner cracks [8]. Their approach achieved an accuracy of 97%, emphasizing the importance of incorporating different anchor scales and aspect ratios to handle different crack geometries effectively. By considering a wide range of crack characteristics, the Fast RCNNbased system developed by Li et al. provided robust crack detection capabilities, supporting comprehensive bridge inspection and maintenance.

Additionally, Liu et al. (2020) introduced a hybrid Fast RCNN algorithm that achieved an accuracy of 92.4% [5]. Their approach combined the strengths of different deep learning architectures, leveraging the efficiency of the Fast RCNN while integrating additional contextual information from the image. This hybrid approach improved crack detection efficiency while maintaining competitive accuracy levels, offering a practical solution for real-time bridge inspection applications.

You Only Look Once (YOLO) algorithms have gained popularity for their real-time object detection capabilities, including crack detection in bridges. In 2019, Lu et al. employed a YOLO-based method with an accuracy of 88% [11]. Their research highlighted the algorithm's effectiveness in detecting various types of cracks, including diagonal, corner, and T-shaped cracks. By utilizing a single-stage architecture, the YOLO-based system provided fast and reliable crack detection, enabling efficient bridge inspection in real-time scenarios.

In another study conducted in 2021, Xu et al. applied the YOLOv3 algorithm, achieving an accuracy of 94.2% [10]. Their research focused on detecting cracks of different sizes and shapes, including narrow cracks and those with low contrast. The YOLOv3 algorithm's ability to process images in real-time, coupled with its accuracy in crack detection, provided a valuable tool for bridge inspectors to detect cracks promptly and accurately, facilitating timely maintenance and ensuring structural integrity.

DOI:10.4186/ej.2023.27.12.11

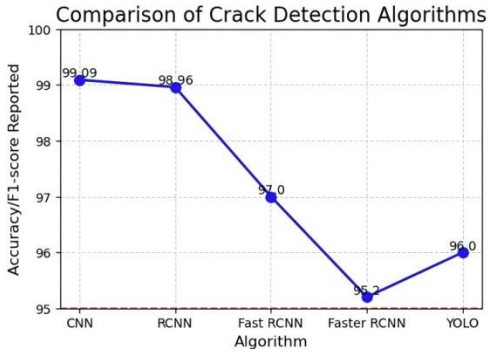


Fig. 1. Comparison graph of deep learning algorithms.

The graph Fig. 1 highlights the advancements in crack detection algorithms over the years, with CNN-based methods consistently achieving high accuracies. The Faster RCNN algorithm also demonstrates significant progress in accuracy, while RCNN and YOLO algorithms show slightly lower but still competitive performance.

## 3. Methodology

### 3.1. Dataset

A representative and typical dataset is very important for the Algorithm for crack detection task. The dataset used in this work contains 40000 images: 20000 crack and 20000 non-cracks with 227x227 pixels with RGB channels. As per the information provided, the dataset is a combination of two datasets to provide sufficient variance amongst the data samples. Dataset contains different types of cracks on concrete structures, samples of the crack and non-crack images. These images are the input to the crack detection algorithm.

### 3.2. Data Preprocessing

In the data preprocessing phase of this study, the image dataset undergoes meticulous organization and preparation. Positive and negative images, representing concrete surfaces with and without cracks, are sourced from specified directories using a custom function named **generate_df**. Subsequently, the dataset is structured into separate DataFrames—**positive_df** and **negative_df**—each containing file paths and corresponding labels. These DataFrames are then concatenated into a comprehensive dataset, denoted as **data**, which is strategically split into training and test sets using the **train_test_split** function from scikit-learn. The training set, a randomized subset comprising 80% of the data (9000 samples for computational efficiency), serves as the foundation for model training.

To augment and preprocess the image data effectively, TensorFlow's powerful **ImageDataGenerator** is employed. Two distinct generators are instantiated: **train_generator** for the training set and **test_generator** for the test set. These generators incorporate essential transformations, including pixel rescaling to a range of [0, 1]. Furthermore, images are seamlessly flowed from the DataFrames into the model using the **flow_from_dataframe** method, ensuring uniformity in size (120x120 pixels), color representation (RGB), and batch processing (32 images per batch). The training images are intentionally shuffled, with a seed set for reproducibility purposes. These preprocessing steps collectively establish a well-structured and augmented dataset, laying the groundwork for the subsequent training of a CNN and RCNN for binary image classification in the realm of crack detection. Then, these meticulously prepared images are provided as input to the CNN and RCNN during the training phase, enabling the model to learn and discern patterns associated with cracks and non-cracks for robust classification.

To create the CNN & RCNN we use the Keras library, which is known to be the most popular library by the deep learning community that allows transparently using TensorFlow, the library developed by Google for deep learning. Other libraries like matplotlib and Seaborn have been used for visualizations and attractive statistical graphics.

### 3.3. Convolutional Neural Network classification

The suitability of the CNN architecture employed in the code for crack detection is grounded in its design, specifically tailored to effectively address the nuances of the crack detection problem. The architecture comprises multiple layers, strategically incorporating convolutional layers, pooling layers, and fully connected layers. Each layer plays a crucial role in the overall functionality of the network.

The initial input layer serves as the gateway for image data, allowing seamless integration of information into the network. Subsequent convolutional and pooling layers work in tandem to systematically process the input, extracting pertinent features essential for accurate crack detection. The architecture is adept at capturing spatial patterns within the images, a critical aspect of crack identification.

The output layer, employing a sigmoid activation function, serves the pivotal role of predicting the presence or absence of cracks based on the processed features. This binary outcome aligns with the nature of the crack detection task, simplifying the decision-making process.

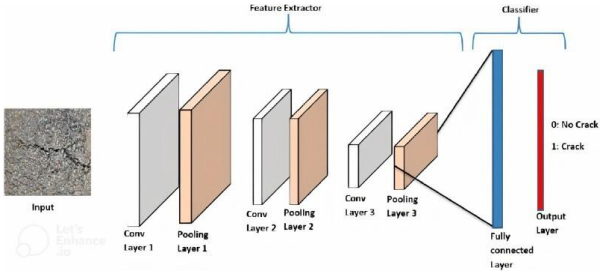


Fig. 2. Architecture of the proposed network CNN.

Table 1 shows the parameters of the proposed architecture which consists of two convolutional layers and two dense layers, as well as polling operations after each convolutional layer. The "Kernel Size" column shows the size of the convolutional kernel for each Conv2D layer. The "Kernel #" column shows the number of kernels (filters) for each Conv2D layer. The "Param. #" column shows the number of trainable parameters for each layer.

Table 1. Parameters of each layer for proposed CNN.

| Layer Type | Kernel Size | Kernel # | Param. # |
|---|---|---|---|
| Input | - | - | 0 |
| Conv2D (filters=16, kernel=3) | 3*3*3 | 16 | 448 |
| MaxPooling2D (Pool=2) | 2*2*1 | - | 0 |
| Conv2D (filters=32, kernel=3) | 3*39316 | 32 | 4640 |
| MaxPooling2D (Pool=2) | 2*2*1 | - | 0 |
| GlobalAvgPooling2D | - | - | 0 |
| Dense (units=128) | - | - | 16512 |
| Dense (units=128) | - | - | 16512 |
| Dense (units=1) with sigmoid | - | - | 129 |

Overall, the above architecture, with its well-defined layers and parameter choices, is tailored to efficiently discern and process intricate patterns within images, making it highly suitable for the crack detection problem at hand. The presented parameters underscore the meticulous consideration given to network configuration, reinforcing its efficacy in addressing the challenges posed by crack detection.

The stepwise algorithm used for implementation of CNN is as follows:

1. Define the layers of the convolutional neural network (CNN) model using the Keras functional API.
2. Create the model by specifying the inputs and outputs.
3. Compile the model with the Adam optimizer, binary cross-entropy loss, and accuracy metric.
4. Train the model using the 'fit' function, specifying the training and validation data, number of epochs, and an early stopping callback to prevent overfitting.
5. Visualize the training and validation loss over time using Plotly.
6. Predict the labels for the test images using the trained model.
7. Define a function, 'evaluate_model', to evaluate the model's performance on the test set. This includes calculating the test loss, accuracy, and generating a confusion matrix and classification report.
8. Call the 'evaluate_model' function with the trained model.
9. Display sample test images with their predicted labels, highlighting correct and incorrect predictions.
10. Identify the indices of the misclassified images.
11. Display the misclassified images from the test set.

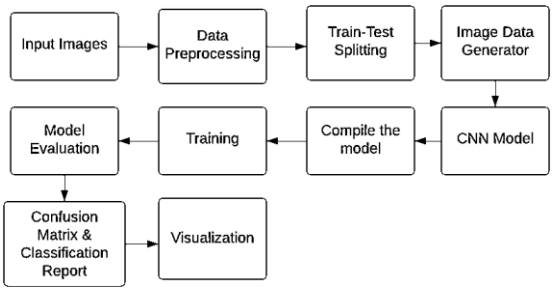


Fig. 3. Block diagram for CNN.

### 3.4. Region Based Convolutional Neural Network Classification

The RCNN architecture proposed in this paper consists of region proposal and convolutional neural networks. It starts by generating region proposals using selective search or a similar algorithm. These proposals are then fed into a CNN, which extracts features from each region. The extracted features are used to classify and localize objects within the regions. The RCNN architecture effectively combines the power of CNNs with region-based processing, making it suitable for accurate crack detection tasks.

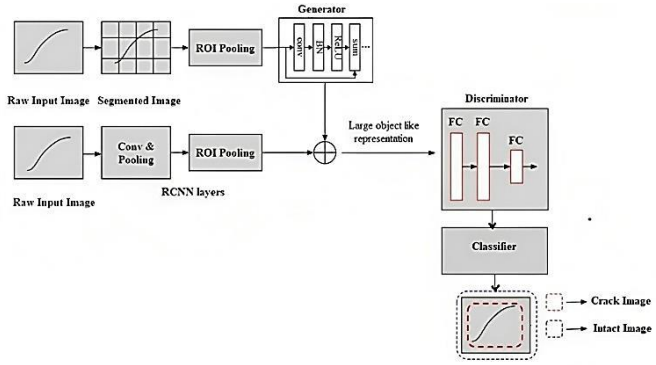


Fig. 4. Architecture of the proposed network RCNN.

The RCNN model consists of a base network with two convolutional layers and two max pooling layers, followed by a Region Proposal Network (RPN) and a Region of Interest (ROI) Pooling layer. The RPN consists of a single convolutional layer with 512 filters and a 1x1 convolutional layer with sigmoid activation for objectness classification and linear activation for bounding box regression, respectively. The ROI Pooling layer uses a time-distributed max pooling operation with 7x7 pooling regions to extract features from each region of interest (ROI). The pooled features are then flattened and passed through two fully connected layers with 128 units each and ReLU activation. Finally, the RCNN model has two output layers, one for classification and one for regression, each with a time-distributed dense layer with softmax and linear activation, respectively.

Table 2. Parameters of each layer for proposed RCNN.

| Layer | Kernel Size | Kernel # | Param. # |
|---|---|---|---|
| Conv1 | 3x3x3 | 32 | 896 |
| Pool1 | 2x2x1 | - | 0 |
| Conv2 | 3x3x3 | 64 | 18496 |
| Pool2 | 2x2x1 | - | 0 |
| Conv3 | 3x3x3 | 128 | 73856 |
| Pool3 | 2x2x1 | - | 0 |
| Conv4 | 3x3x3 | 256 | 295168 |
| Pool4 | 2x2x1 | - | 0 |
| Flatten | - | - | 0 |
| Dense1 | - | 512 | 3277312 |
| Dense2 | - | 1 | 513 |

The stepwise algorithm used for implementation of RCNN is as follows:

1. Define two functions: 'get_rpn_layer' and 'get_roi_layer', which will be used to build the Region Proposal Network (RPN) and ROI (Region of Interest) pooling layer, respectively.
2. Define the main RCNN model using the 'get_rcnn_model' function. This model consists of input layers, shared convolutional layers, the RPN layer, and the ROI pooling layer.
3. Create an instance of the RCNN model.
4. Compile the model with the Adam optimizer, binary cross-entropy loss, and accuracy metric.
5. Train the model using the `fit` function. Pass the training and validation data generators as inputs and specify the number of epochs to train. Use the 'EarlyStopping' callback to stop training early if the validation loss doesn't improve for 3 consecutive epochs.
6. Plot the training and validation loss over time using Plotly Express.
7. Make predictions on the test set using the trained model.
8. Define a function, 'evaluate_model', to evaluate the model's performance on the test set. This includes calculating the test loss, accuracy, and generating a confusion matrix and classification report. Call the 'evaluate_model' function with the trained model.
9. Display sample test images with their predicted labels, highlighting correct and incorrect predictions.
10. Identify the indices of the misclassified images.

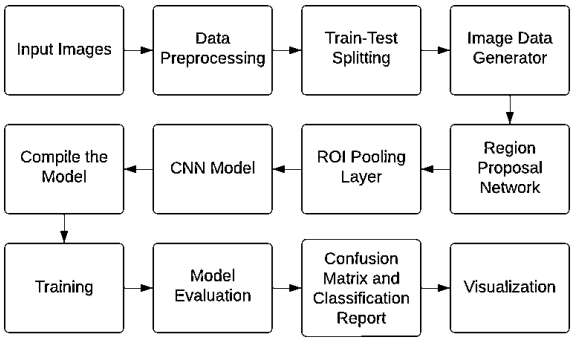


Fig. 5. Block diagram for RCNN Model.

### 3.5. Evaluation Metrics

In this section, we outline a comprehensive evaluation of the deep learning models employed in our paper, focusing on crack detection using Convolutional Neural Networks (CNN) and Recurrent Convolutional Neural Networks (RCNN). The choice of metrics aims to provide a multifaceted assessment of the models' effectiveness.

A. Accuracy is a commonly used metric that measures the proportion of correctly classified samples to the total number of samples. It can be calculated using the following equation:

$$\text{Accuracy} = \frac{\text{No. of correctly classified samples}}{\text{Total number of samples}}$$

B. Test loss quantifies the discrepancy between the predicted output and the actual ground truth values during the testing phase. It provides insights into the model's overall performance. Lower test loss values indicate better model performance.

C. The number of epochs refers to the number of times the entire dataset is passed through the model during training. It reflects the number of

iterations needed for the model to converge and achieve optimal performance.

D. The total number of samples used denotes the size of the dataset employed for training, validation, and testing purposes. A larger dataset often leads to more reliable and accurate model performance.

E. Training and validation loss are Vital metrics used to monitor the model's performance during the training phase. Training loss represents the error between predicted and actual values during the training process, while validation loss indicates the error during the validation phase. These metrics help in assessing the model's generalization capabilities and identifying possible overfitting or underfitting issues.

F. A confusion matrix is a tabular representation that provides a detailed analysis of a classifier's performance. It summarizes the true positive, true negative, false positive, and false negative predictions. From the confusion matrix, several evaluation metrics can be derived, including precision, recall, and F1 score.

G. Precision quantifies the proportion of true positive predictions out of the total positive predictions, emphasizing the model's accuracy in positive identifications:

$$\text{Precision} = \frac{TP}{TP + FP}$$

H. Recall, also known as sensitivity or true positive rate, measures the proportion of actual positive instances correctly classified by the model. It is computed using the following equation:

$$\text{Recall} = \frac{TP}{TP + FN}$$

I. F1 score combines precision and recall into a single metric that balances both metrics. It provides a harmonic mean of precision and recall, allowing for a comprehensive assessment of the model's overall performance:

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

These evaluation metrics, along with the corresponding results [4], are presented in detail through tables and figures, providing a comprehensive analysis of the effectiveness of our CNN and RCNN models in crack detection.

## 4. Results & Discussions

Based on the research, a Convolutional Neural Network (CNN) achieved a maximum accuracy of 98% and RCNN achieved a maximum accuracy of 96%, for detecting cracks, as reported in appendix section. However, in an effort to further improve the accuracy rate and enhance detection results, certain parameters like Epochs (3.5[C]), Number of samples for training (3.5[D]), in the CNN & RCNN architecture were modified, as shown in Table 3(a) and Table 3(b).

Table 3. (a) Changed Parameters of CNN.

| Accuracy % | Test loss | Epoch | No. of Samples |
|---|---|---|---|
| **98.37%** | 0.0454 | 500 | 6000 |
| **98.48%** | 0.0414 | 600 | 7000 |
| **98.98%** | 0.0290 | 700 | 8000 |
| **99.44%** | 0.0196 | 700 | 9000 |

Table 3. (b) Changed Parameters of RCNN.

| Accuracy % | Test loss | Epoch | No. of Samples |
|---|---|---|---|
| **97.87** | 0.0465 | 500 | 6000 |
| **98.48** | 0.0414 | 600 | 7000 |
| **98.69** | 0.0401 | 700 | 8000 |
| **98.11** | 0.0318 | 700 | 9000 |

Based on the parameters we tested, we were able to achieve a high level of accuracy ranging from 98% to 99.25%, for CNN architecture as shown in the graph Fig. 5(a). Similarly we were able to achieve accuracy ranging from 97% to 98% as shown in graph Fig. 5(b) for RCNN architecture. The different splits and sample sizes

DOI:10.4186/ej.2023.27.12.11

did not have a significant impact on the accuracy (3.5[A]), but it's worth noting that the model performed slightly better with a larger sample size. Additionally, as the test loss (3.5[B]) decreased, the accuracy increased, which is to be expected. Overall, we were able to obtain a high level of accuracy with our model for CNN & RCNN using the tested parameters.

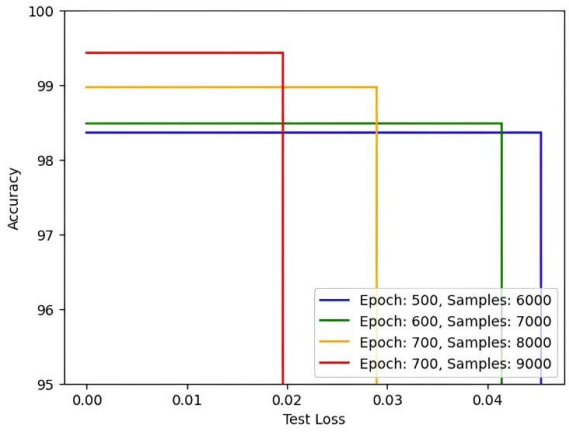


Fig 5. (a) Accuracy vs Test Loss Graph for CNN.

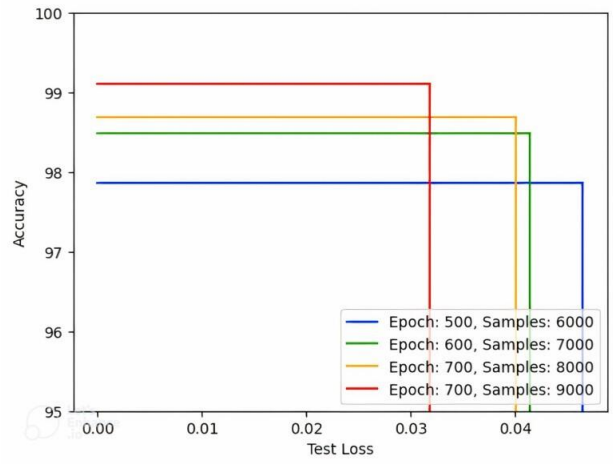


Fig 5. (b) Accuracy vs Test Loss Graph for RCNN.

After extensive testing, we were able to achieve a maximum accuracy of 99.25% for CNN architecture and 99.00% for RCNN architecture. This was accomplished using a training dataset of 8000 samples and training for 800 epochs with an 80:20 split between the training and validation sets. The Training and validation loss (3.5[E]) are depicted in the accompanying Fig. 6(a) for CNN and Fig. 6(b) for RCNN.

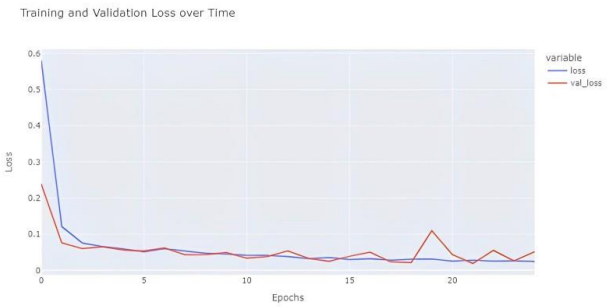


Fig 6. (a) Training and Validation Loss Graph for CNN.

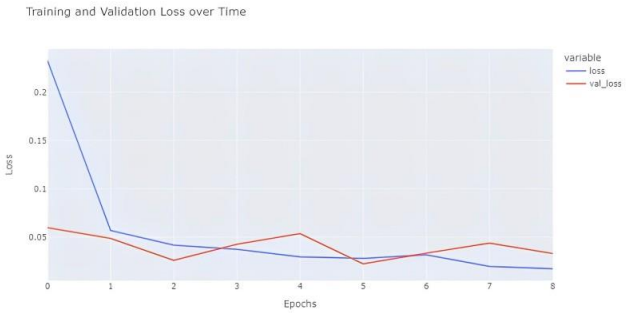


Fig 6. (b) Training and Validation Loss Graph for RCNN.

The next tables represent a confusion matrix(3.5[F]) for crack detection. Table 4(a) (CNN) and Table 4(b) (RCNN) outline the model's performance, distinguishing between true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). TP indicates correct crack identification, TN is accurate non-crack identification, FP signals false alarms, and FN represents missed crack detections. These matrices gauge the models' accuracy and reveal insights into their crack detection abilities, visualized through heatmaps.

In crack detection, false positives (FP) could trigger unnecessary maintenance, impacting costs and safety. Meanwhile, false negatives (FN) pose more severe risks, indicating overlooked cracks and compromising safety. Achieving a balance is vital for an effective system. Evaluating heatmaps adds depth, emphasizing the practical consequences. This nuanced approach highlights the need for minimizing both false positives and false negatives to enhance the reliability of crack detection systems.

Table 4. (a) Confusion Matrix Table For CNN.

| | | Detection | |
|---|---|---|---|
| | | TRUE | FALSE |
| Actual | TRUE | 902 | 7 |
| | FALSE | 3 | 888 |

Table 4(b). Confusion Matrix Table For RCNN.

| | | Detection | |
|---|---|---|---|
| | | TRUE | FALSE |
| Actual | TRUE | 907 | 2 |
| | FALSE | 14 | 877 |

Here Table 5(a) and Table 5(b) show the comparison of existing and proposed values of classification report for each of the algorithms used, that is CNN and RCNN respectively. The classification report for crack detection using CNN and RCNN provides a comprehensive summary of the model's performance metrics, including recall, precision, and F1 score. Precision (3.5[G]) quantifies the model's accuracy in detecting cracks, Recall (3.5[H]) measures the model's ability to correctly identify cracks, and the F1 score (3.5[I]) balances both precision and recall. We can observe that the results of both the implemented algorithms have improved as compared to the existing results.

Table 5. (a) Comparison table for CNN.

| CNN | Recall | Precision | F1-score |
|---|---|---|---|
| **Existing** | 0.80 | 0.85 | 0.82 |
| **Proposed** | 1.00 | 0.99 | 0.99 |

Table 5. (b) Comparison table for RCNN.

| RCNN | Recall | Precision | F1-score |
|---|---|---|---|
| **Existing** | 0.89 | 0.83 | 0.86 |
| **Proposed** | 1.00 | 0.98 | 0.99 |

The enhanced recall, precision, and F1 score values in Table 5(a) and Table 5(b) signify substantial improvements in the proposed model over existing results, showcasing its heightened efficacy in safety-critical applications like crack detection. A recall of 1.00 indicates a minimized rate of false negatives, crucial for ensuring the accurate identification of all crack instances in safety scenarios. The precision values of 0.99 for CNN and 0.98 for RCNN demonstrate a notable reduction in false positives, contributing to a more reliable crack detection system. The F1 score of 0.99 for both CNN and RCNN reflects a balanced approach, minimizing missed detections and false alarms.

These are some sample results we get for our two implemented algorithms, which detect if the given input image has a crack or not.

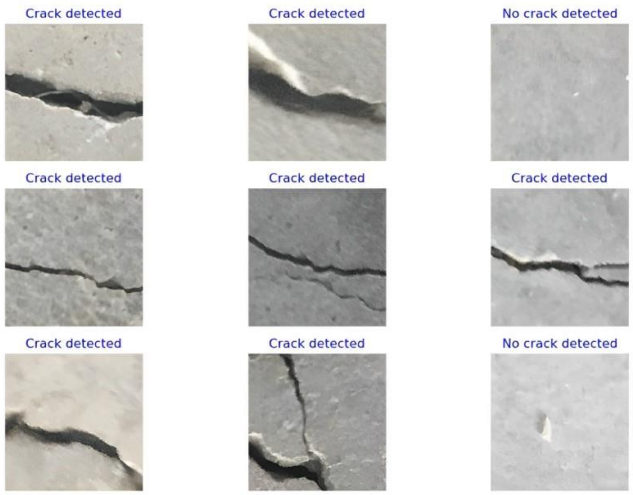


Fig. 6. Result of the Proposed method.

## 5. Conclusion

Our research stands as a testament to the effectiveness of CNN in the domain of crack detection from images obtained through UAVs. This conclusion is rooted in a meticulous exploration of numerical data, unraveling key insights into the comparative performance of two prominent deep learning models, CNN and RCNN.

The standout numerical finding lies in the remarkable accuracy achieved by the CNN model, reaching an impressive 99.44%, overshadowing the RCNN model's performance at 99.11%. This substantial accuracy gap forms a robust numerical foundation, unequivocally establishing the supremacy of the CNN model in the specific task of UAV-based crack detection.

Moreover, to fortify the reliability and robustness of our conclusions, we conducted comprehensive parameter tuning experiments. The CNN model consistently outperformed the RCNN model across various experiments, underscoring the CNN model's superior performance in crack detection. The CNN and RCNN models exhibit superior accuracy compared to other algorithms such as YOLO and Fast RCNN, which report accuracies ranging from 95% to 98%.

Further substantiating our findings, extensive parameter tuning experiments were undertaken, offering a nuanced understanding of model behavior under diverse conditions. The resulting numerical outcomes

consistently favored the CNN model, showcasing its resilience and superior performance across various parameter configurations. These numerical details serve as pivotal support, enhancing the reliability of our overarching conclusion.

Our unwavering assertion is that the CNN model emerges as a more robust and reliable solution for UAV-captured crack detection, distinctly validated by its superior accuracy. This conclusion is underlined by the consistency of CNN's outperformance over RCNN across a spectrum of experimental scenarios, adding a quantitative dimension to our claim.

The significance of our numerical findings extends beyond the confines of our research, holding potential implications for industries reliant on precise crack detection systems, notably in civil engineering and infrastructure maintenance. The stark numerical superiority of the CNN model, with an accuracy of 99.44%, underscores its potential to avert damages and bolster safety through early crack detection.

Looking forward, our future research trajectory is guided by these numerical revelations. Delving into advanced architectures like DenseNet and ResNet, backed by their promising track records in computer vision tasks, aims to elevate the accuracy and robustness of our model. Additionally, a strategic augmentation of our dataset, encompassing diverse crack images captured in varied conditions, aims to fortify real-world performance.

In summation, our comprehensive numerical exploration fortifies the conclusion that the CNN model, with its exceptional accuracy and consistent performance, stands as a potent solution for UAV-based crack detection. This conclusion resonates with the data-driven essence of our research, providing a nuanced and quantitative perspective on the efficacy of our proposed model.

## References


[1] F.-C. Chen and M. R. Jahanshahi, "NB-FCN: Real-time accurate crack detection in inspection videos using deep fully convolutional network and parametric data fusion," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 8, pp. 5325–5334, 2020.

[2] C. Chen, H. Seo, and Y. Zhao, "A novel pavement transverse cracks detection model using WT-CNN and STFT-CNN for smartphone data analysis," *Int. J. Pavement Eng.*, vol. 23, no. 12, pp. 4372–4384, 2022.

[3] S. Li and X. Zhao, "Image-based concrete crack detection using convolutional neural network and exhaustive search technique," *Adv. Civ. Eng.*, vol. 2019, pp. 1–12, 2019.

[4] Y. Ding et al., "Crack identification method of steel fiber reinforced concrete based on deep learning: A comparative study and shared crack database," *Adv. Mater. Sci. Eng.*, vol. 2021, pp. 1–10, 2021.

[5] S. Li, Y. Cao, and H. Cai, "Automatic pavement-crack detection and segmentation based on steerable matched filtering and an active contour model," *J. Comput. Civ. Eng.*, vol. 31, no. 5, p. 04017045, 2017.

[6] B. Ramalingam et al., "Visual inspection of the aircraft surface using a teleoperated reconfigurable climbing robot and enhanced deep learning technique," *Int. J. Aerosp. Eng.*, vol. 2019, pp. 1–14, 2019.

[7] K. Hacıefendioğlu and H. B. Başağa, "Concrete road crack detection using deep learning-based faster R-CNN method," *Iran. J. Sci. Technol. Trans. Civ. Eng.*, vol. 46, no. 2, pp. 1621–1633, 2022.

[8] X. Xu et al., "Crack detection and comparison study based on Faster R-CNN and Mask R-CNN," *Sensors (Basel)*, vol. 22, no. 3, p. 1215, 2022.

[9] D. Kang, S. S. Benipal, D. L. Gopal, and Y.-J. Cha, "Hybrid pixel-level concrete crack segmentation and quantification across complex backgrounds using deep learning," *Autom. Constr.*, vol. 118, no. 103291, p. 103291, 2020.

[10] S. Li et al., "Detection of concealed cracks from ground penetrating radar images based on deep learning algorithm," *Constr. Build. Mater.*, vol. 273, no. 121949, p. 121949, 2021.

[11] S. Lu, B. Wang, H. Wang, L. Chen, M. Linjian, and X. Zhang, "A real-time object detection algorithm for video," *Comput. Electr. Eng.*, vol. 77, pp. 398–408, 2019.

[12] H. Kim, S. H. Sim, and S. Cho. "Unmanned aerial vehicle (UAV)-powered concrete crack detection based on digital image processing." Illinois.edu. Accessed: 12 Jun 2023. [Online]. Available: http://sstl.cee.illinois.edu/papers/aeseancrisst15/194_Kim_Unmanned.pdf

[13] S. Murao, Y. Nomura, H. Furuta, and C.-W. Kim. "Concrete crack detection using UAV and deep learning." Snu.ac.kr. Accessed: 12 Jun 2023. [Online]. Available: https://s-space.snu.ac.kr/bitstream/10371/153261/1/29.pdf

[14] R. Li, J. Yu, F. Li, R. Yang, Y. Wang, and Z. Peng, "Automatic bridge crack detection using unmanned aerial vehicle and Faster R-CNN," *Constr. Build. Mater.*, vol. 362, no. 129659, p. 129659, 2023.

[15] X. Wu, W. Li, D. Hong, R. Tao, and Q. Du, "Deep learning for UAV-based object detection and tracking: A survey," 2021, *arXiv*: [cs.CV].

[16] A. Ramachandran and A. K. Sangaiah, "A review on object detection in unmanned aerial vehicle surveillance," *International Journal of Cognitive Computing in Engineering*, vol. 2, pp. 215–228, 2021.

[17] J. Lee, J. Wang, D. Crandall, S. Sabanovic, and G. Fox, "Real-time, cloud-based object detection for unmanned aerial vehicles," in *2017 First IEEE International Conference on Robotic Computing (IRC)*, 2017, pp. 36–43.

[18] S.-C. Han, W. Shen, and Z. Liu, "Deep Drone: Object detection and tracking for smart drones on embedded system," 2016.
[19] R. Gandhi. "R-CNN, fast R-CNN, faster R-CNN, YOLO—Object detection algorithms." Towards Data Science. Accessed: 12 Jun 2023. [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e
[20] B. P. Babu and I. Vairavasundaram, "Analysis of power quality in a grid system connected with a three phase induction motor," *International Journal of Emerging Electric Power Systems*, vol. 21, no. 4, pp. 20200054, 2020.
[21] A. Rahai, M. Rahai, M. Iraniparast and M. Ghatee, "Surface crack detection using deep convolutional neural network in concrete structures," in *2022 IEEE 5th International Conference on Image Processing Applications and Systems (IPAS)*, Genova, Italy, 2022, pp. 1-5, doi: 10.1109/IPAS55744.2022.10052790.

**Bandla Pavan Babu** received the B.Tech. Degree in Electrical and Electronics Engineering from JNTUA in 2011. He received the M.Tech. Degree in Electrical Power Systems from JNTUA, India, in 2014, and a PhD in Electrical Engineering from Vellore Institute of Technology, Vellore in 2022. He served as an Assistant Professor over 10 years in premier institutions and currently working as Assistant Professor in School of Computing Science and Engineering at VIT Bhopal University, Madhya Pradesh, India. His research area is Power Quality, Electric Vehicle Technology, Smart Grid and Renewable Energy Sources, Artificial Intelligence and Machine Learning Techniques.

**Sarah Khandagale** was born in Lonavala, India, on April 27,2001. She pursued her education in Computer Science and Engineering ,earning a Bachelor of Technology (B Tech) degree with specialization in Artificial Intelligence and Robotics from D Y Patil International University, Akurdi, Pune, Maharashtra, India in 2023. She gained valuable work experience as a Software Developer Intern at Cppsecrets Technologies Pvt. Ltd. where she published technical articles related to python.

**Vedashree Shinde** was born in Pune, India, on December 10 ,2000. She pursued her education in Computer Science and Engineering, earning a Bachelor of Technology (B Tech) degree with specialization in Artificial Intelligence and Machine learning from D Y Patil International University, Akurdi, Pune, Maharashtra, India in 2023. She gained valuable work experience as a Software Developer Intern at Excellarate Softech Pvt. Ltd. Where she contributed to developing a virtual assistant for online retailers on Telegram.

**Saurabh Gargote** was born in Pune, India, on July 19,2000. He pursued her education in Computer Science and Engineering, earning a Bachelor of Technology (B Tech) degree with specialization in Artificial Intelligence and Machine learning from D Y Patil International University, Akurdi, Pune, Maharashtra, India in 2023. He gained valuable work experience as Software Developer Intern at Wajooba India Pvt. Ltd. Where he developed production-grade applications in multiple technologies -Node.js, Dart, Flutter.

**Kishore Bingi** received the B.Tech. Degree in Electrical and Electronics Engineering from Acharya Nagarjuna University, India, in 2012. He received the M.Tech. Degree in Instrumentation and Control Systems from the National Institute of Technology Calicut, India, in 2014, and a PhD in Electrical and Electronic Engineering from Universiti Teknologi PETRONAS, Malaysia, in 2019. From 2014 to 2015, he worked as an Assistant Systems Engineer at TATA Consultancy Services Limited, India. From 2019 to 2020, he worked as Research Scientist and Post-Doctoral Researcher at the Universiti Teknologi PETRONAS, Malaysia. From 2020 to 2022, he served as an Assistant Professor at the Process Control Laboratory, School of Electrical Engineering, Vellore Institute of Technology, India. Since 2022 he has been working as a faculty member at the Department of Electrical and Electronic Engineering at Universiti Teknologi PETRONAS, Malaysia. His research area is developing fractional-order neural networks, including fractional-order systems and controllers, chaos prediction and forecasting, and advanced hybrid optimization techniques. He is an IEEE and IET Member and a registered Chartered Engineer (CEng) from Engineering Council UK. He serves as an Editorial Board Member for the International Journal of Applied Mathematics and Computer Science and Academic Editor for Mathematical Problems in Engineering and the Journal of Control Science and Engineering.

DOI:10.4186/ej.2023.27.12.11

## Appendix

The following table summarizes the performance of various deep learning algorithms for crack detection in civil structures. The accuracy percentages and corresponding studies showcase the advancements in the field.

Table 7. Comparison table of deep learning algorithms.

| Algorithm | Accuracy% | Study |
|---|---|---|
| CNN | 99.06 | [3] Li et al. (2019) |
| | 97.20 | [2]2021 |
| | 98.00 | [1] Chen et. Al (2021) |
| RCNN | 96.20 | [6] (2016) |
| | 92.4 | [5] (2017) |
| | 95.78 | [4] Dong et al. (2021) |
| Faster RCNN | 93.40 | [7] Yang et al. (2018) |
| | 95.00 | [9] Liu et al. (2020) |
| | 97.00 | [9] Li et al. (2022) |
| YOLO | 88.00 | [11] Lu et al. (2019) |
| | 94.20 | [10] Xu et al. (2021) |