iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.83081

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Bridge Crack Detection Using Convolutional Neural Networks

Kavish Gupta[1], Chander Dev Singh[2], Saksham Choudhary[3], Yash Jogdand[4]
Pune Vidyarthi Griha's College of Engineering and Technology, Pune, India

Abstract: ***Every day, millions of people drive over bridges without thinking twice about what's holding them up. But for civil engineers and infrastructure authorities, bridge health is a constant and serious concern. These structures age under relentless stress — traffic loads, temperature swings, corrosion, and moisture all chip away at the concrete over time. Cracking is usually the earliest visible sign that something is starting to go wrong, and catching it early is the difference between a quick repair and a major structural failure. Traditional inspection relies on engineers physically walking and examining bridges — a process that is slow, subjective, and sometimes dangerous. In this paper, we present a CNN-based automated crack detection system that learns to distinguish cracked from intact concrete surfaces directly from images. Trained on a large labeled dataset, our model achieved a classification accuracy of 93.22 percentage, showing that deep learning can meaningfully automate this part of structural inspection.***

Keywords: Bridge Crack Detection; Deep Learning; CNN; Structural Health Monitoring

I. INTRODUCTION

Bridges are everywhere, and most of us cross them without giving a second thought to what keeps them standing. But for engineers and city planners, the health of bridge infrastructure is a constant concern. These structures carry enormous loads day after day, year after year, and the wear eventually shows. One of the earliest and most telling signs of that wear is cracking.

Cracks in concrete bridges don't always start out dangerous. A hairline fracture here, a small surface split there — on their own, they might seem harmless. But left unchecked, cracks grow. They let in water, accelerate corrosion of the reinforcing steel inside, and can eventually threaten the structural stability of the entire bridge. That is why catching them early matters so much.

The traditional way of doing this involves sending inspectors out to physically examine the bridge — climbing underneath it, scanning surfaces with flashlights, and noting any damage they spot. It is slow, expensive, and subject to human error. Two inspectors looking at the same crack might assess it very differently.

That is the problem we wanted to address in this project. We asked: can a computer learn to spot cracks the way a trained engineer does — or even better? With advances in deep learning, particularly Convolutional Neural Networks (CNNs), the answer turns out to be yes. In this paper, we describe a CNN-based system we built and tested to automatically detect cracks in bridge surface images, and the results were encouraging.

II. MATERIALS AND METHODS

A. Dataset Description

To train and test our model, we used the Concrete Crack Image Dataset available on Kaggle. It is a well-known benchmark in this area of research, widely used for evaluating crack detection models, which also makes it easier to compare our results with other published work.

What makes this dataset useful is not just the volume of images, but the variety. The photos were taken under different lighting conditions, with varying levels of shadow, and on surfaces with different textures. That kind of diversity is important — a model trained only on perfect, well-lit images would likely struggle in real-world inspection conditions where nothing is ideal.

The images are divided into two groups: surfaces that show cracks, and surfaces that do not. This binary setup framed our problem clearly as a supervised classification task, giving the CNN a well-defined target to learn from.

B. Data Preprocessing

Before any image went into the model, we ran it through a preprocessing pipeline. This step is sometimes glossed over in writeups, but it genuinely matters. Feeding raw, inconsistent images into a neural network leads to poor and unstable training.

©IJRASET: All Rights are Reserved

We resized every image to 120×120 pixels — small enough to keep training manageable on our hardware, but large enough that cracks remain clearly visible. After resizing, we normalized the pixel values to bring them into a consistent numerical range. This keeps gradient updates well-behaved during training and prevents the model from being thrown off by lighting differences between images.

III. PROPOSED METHODOLOGY

A. System Workflow

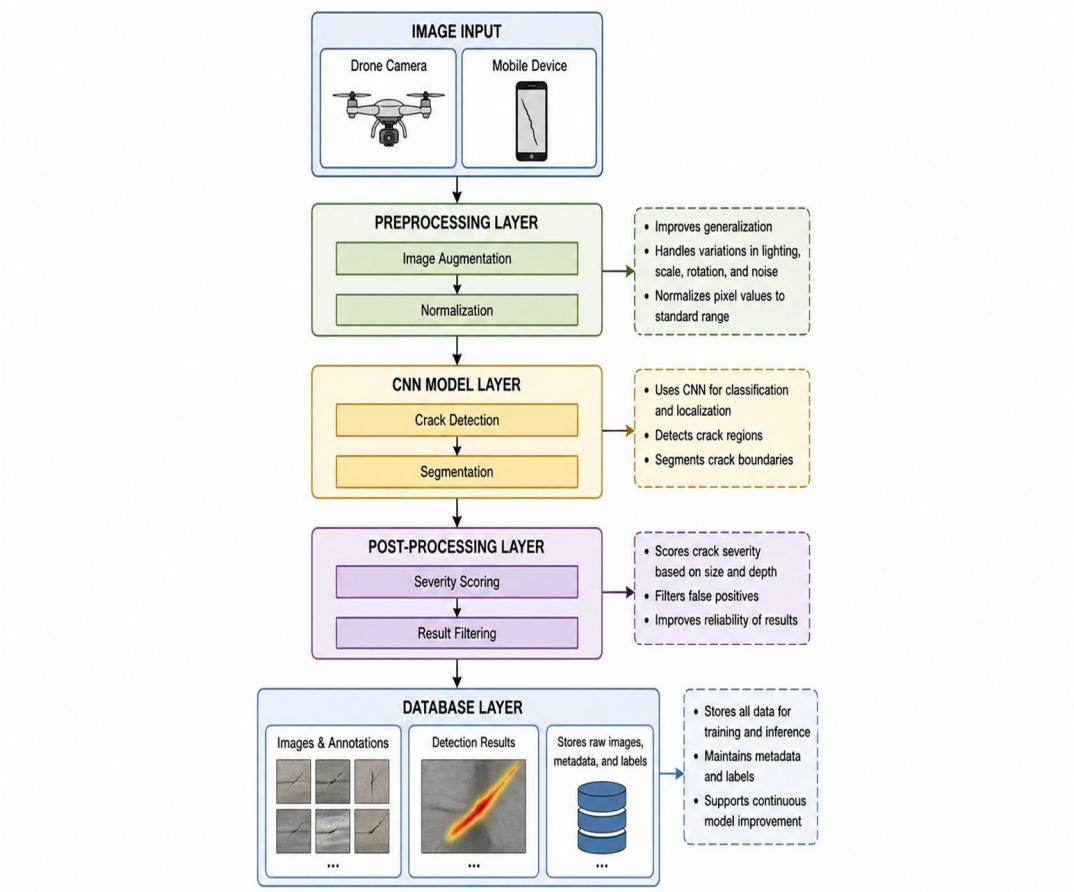


Fig. 1. Workflow of the Bridge Crack Detection System

Figure 1: Workflow of the Bridge Crack Detection System

B. CNN Architecture

The CNN is the core of the system. At a high level, it works by learning — from thousands of labeled examples — which visual patterns correspond to cracks. You do not write rules for it; the network figures them out on its own through training.

Our architecture takes a 120×120 pixel input. After normalization, the image passes through a series of convolutional layers. Think of each layer as a set of filters scanning the image for patterns. Early filters pick up simple features: edges, brightness gradients, surface texture. Deeper filters combine those simple observations into more complex ones, eventually encoding the irregular, elongated shapes that are characteristic of concrete cracks.

Between convolutional layers, we placed max pooling layers. These progressively shrink the feature maps, keeping the computation manageable and forcing the model to retain only the most important information rather than every single pixel.

After the final pooling step, the feature maps are flattened into a vector and passed through fully connected dense layers. These layers combine all the extracted features into a final classification decision. The output is a single sigmoid activation — a probability between 0 and 1. Above 0.5 means crack; below 0.5 means no crack.

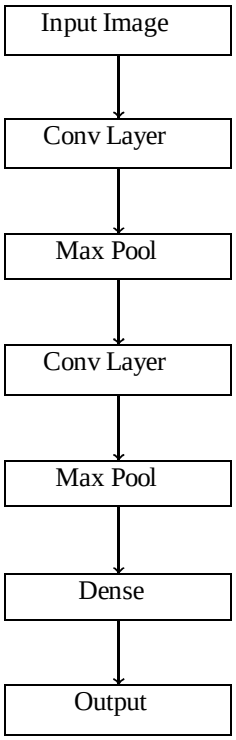


Figure 2: CNN Architecture for Crack Detection

IV. EXPERIMENTAL RESULTS

We trained our model using the Kaggle Concrete Crack Dataset by splitting the images into training, validation, and testing sets. Each set had its own purpose. The training data helped the CNN learn what crack patterns look like and how they differ from normal concrete surfaces. The validation set allowed us to keep track of the model's learning process and check whether it was starting to memorize the training images instead of actually learning useful patterns. The testing set was kept separate throughout training so we could get a fair idea of how the model would perform on completely new images.

We trained the network for 20 epochs with a batch size of 32. The Adam optimizer was chosen because it is known for providing stable and efficient training in deep learning tasks. Since our work focused on identifying only two categories — crack and non-crack images — binary cross-entropy was used as the loss function. During training, the model improved steadily without major fluctuations, which showed that learning was happening in a stable way. Accuracy increased over time while loss gradually reduced, indicating that the network was learning meaningful visual features from the dataset. At the end of training, the model achieved a testing accuracy of 93.22

A. Confusion Matrix

Accuracy alone does not tell the full story, so we examined the confusion matrix to understand exactly where the model gets things right and wrong.

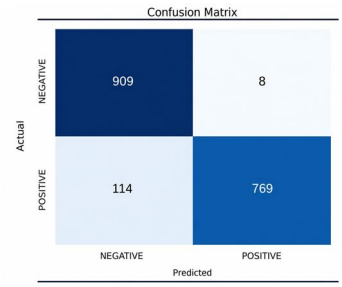


Figure 3: Confusion Matrix

The values in the confusion matrix indicate that most of the images are correctly classified. The number of true negatives and true positives is significantly higher than the number of false predictions, which demonstrates the effectiveness of the model.

Actual / Predicted	Negative	Positive
Negative	909	8
Positive	114	769

Table 1: Confusion Matrix Values

As shown in Table 1, the model correctly identified 909 non-crack images (true negatives) and 769 crack images (true positives). There were 8 false positives and 114 false negatives. The false negatives are the more safety-relevant number — missing a crack is more dangerous than raising a false alarm. At 114 out of 883 actual crack cases, there is room to improve, but it is a reasonable starting point..

B. Performance Metrics

We calculated accuracy, precision, recall, and F1-score for a complete view of model performance. Accuracy is the headline number. Precision tells us how often the model is right when it flags a crack. Recall tells us what fraction of real cracks were detected. F1-score balances the two, which is particularly useful in safety-critical settings where both false alarms and missed detections matter.

Accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \quad (1)$$

Precision:

$$Precision = \frac{TP}{TP + FP} \times 100 \quad (2)$$

Recall:

$$Recall = \frac{TP}{TP + FN} \times 100 \quad (3)$$

F1-score:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \times 100 \quad (4)$$

Metric	Value
Accuracy	93.22%
Loss	0.28
Epochs	20
Batch Size	32

Table 2: Model Performance

C. Training Accuracy Graph

The accuracy curve tells a satisfying story. Epoch 1 sat barely above 60

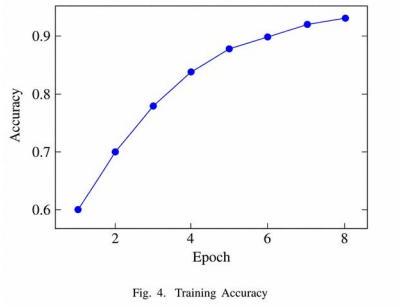


Fig. 4. Training Accuracy

Figure 4: Training Accuracy

D. Training Loss Graph

The loss curve supported what we observed from the accuracy results. At the beginning of training, the loss value was relatively high, starting close to 0.95, which was expected since the model was still learning important crack patterns from the images. As training continued, the loss gradually decreased with each epoch as the Adam optimizer updated the network weights and improved prediction performance. By around epoch 8, the loss had dropped to nearly 0.28, showing that the model was learning effectively from the dataset. One encouraging observation was the smooth behavior of the training process. There were no sudden jumps, unstable fluctuations, or long plateaus in the curve. Instead, the loss reduced in a steady and consistent manner, indicating stable training and suggesting that the model was successfully learning useful features without major optimization problems.

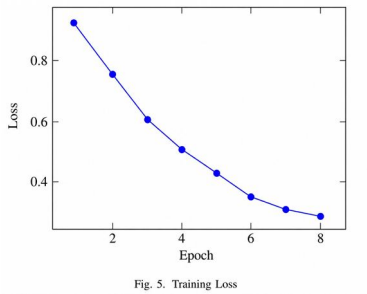


Figure 5: Training Loss

V. ADVANTAGES OF PROPOSED SYSTEM

The main advantage this system has over manual inspection is that it does not get tired, does not have off days, and does not need scaffolding to reach awkward spots on a bridge. Traditional inspection is physically demanding and can be genuinely hazardous, especially on tall or heavily trafficked structures. Speed is another significant benefit. A human inspector might spend hours on a single bridge span. Our system can analyze an image in seconds. Scaled up to a drone survey of a large bridge, inspection times become orders of magnitude faster than walking the structure by hand. There is also the consistency argument. Two different engineers looking at the same crack will sometimes disagree on its severity. A trained model gives the same answer every time for the same input, making it easier to track changes over time and compare results across inspection dates.

VI. CONCLUSION

This project set out to answer a practical question: can a CNN reliably detect cracks in bridge surface images? Based on our results, yes — at least at a level that makes the approach worth taking seriously.

We built and trained a CNN on the Kaggle Concrete Crack dataset and achieved a classification accuracy of 93.22 percent on images the model had never seen. The confusion matrix shows the modelis particularly good at avoiding false alarms (only 8 false positives), though the 114 missed cracks suggest recall could be improved through a larger or more diverse dataset, or a deeper architecture. The system is not ready to replace experienced inspectors outright. Real-world bridge surfaces are messier than a benchmark dataset, and the consequences of a missed crack are serious. But as a first-pass screening tool — something that flags suspicious areas for an engineer to examine more closely — this kind of system has real and immediate potential.

Going forward, we would like to explore transfer learning from pre-trained models like ResNet or VGG, which could push accuracy higher without requiring more training data. We are also interested in integrating this with drone-based capture systems for continuous, large-scale bridge monitoring.

VII. FUTURE WORK

The results so far are promising, but there is a lot of room to build on them. One area we want to explore is drone integration. If you can automate image capture as well as analysis, you have a fully hands-off inspection pipeline — a drone flies the bridge, the CNN processes the footage, and an engineer reviews the flagged sections. That would be a significant step forward for large-scale infrastructure monitoring. We are also interested in real-time detection from live video streams. Right now the system works on static images, but processing video frames continuously would open up new monitoring possibilities — imagine a camera mounted on a maintenance vehicle that flags cracks as it drives across the bridge. On the model side, experimenting with deeper architectures and transfer learning seems like the most direct path to improving accuracy, particularly for reducing the false negative rate, where the safety payoff would be most significant.

VIII. ACKNOWLEDGMENTS

We would like to thank Pune Vidyarthi Griha's College of Engineering and Technology for providing the computing resources and academic environment that made this project possible. Without access to decent hardware and institutional support, the kind of model training we did here simply would not have been feasible.

Thanks also to the faculty members who gave feedback along the way, and to our classmates who pushed back on our ideas often enough that we had to think harder about our choices. That kind of peer pressure is genuinely useful in research.

Finally, we acknowledge the Kaggle Concrete Crack Image Dataset — a well-constructed public resource without which this project would not have been possible. We appreciate the work that went into building and sharing it with the research community.

A. Conflicts of Interest

The authors declare no conflicts of interest. This research was not funded by any external organi-zation, and no outside party had any influence over the study design, data collection, analysis, or reporting. Everything here reflects our own independent work and conclusions.

REFERENCES

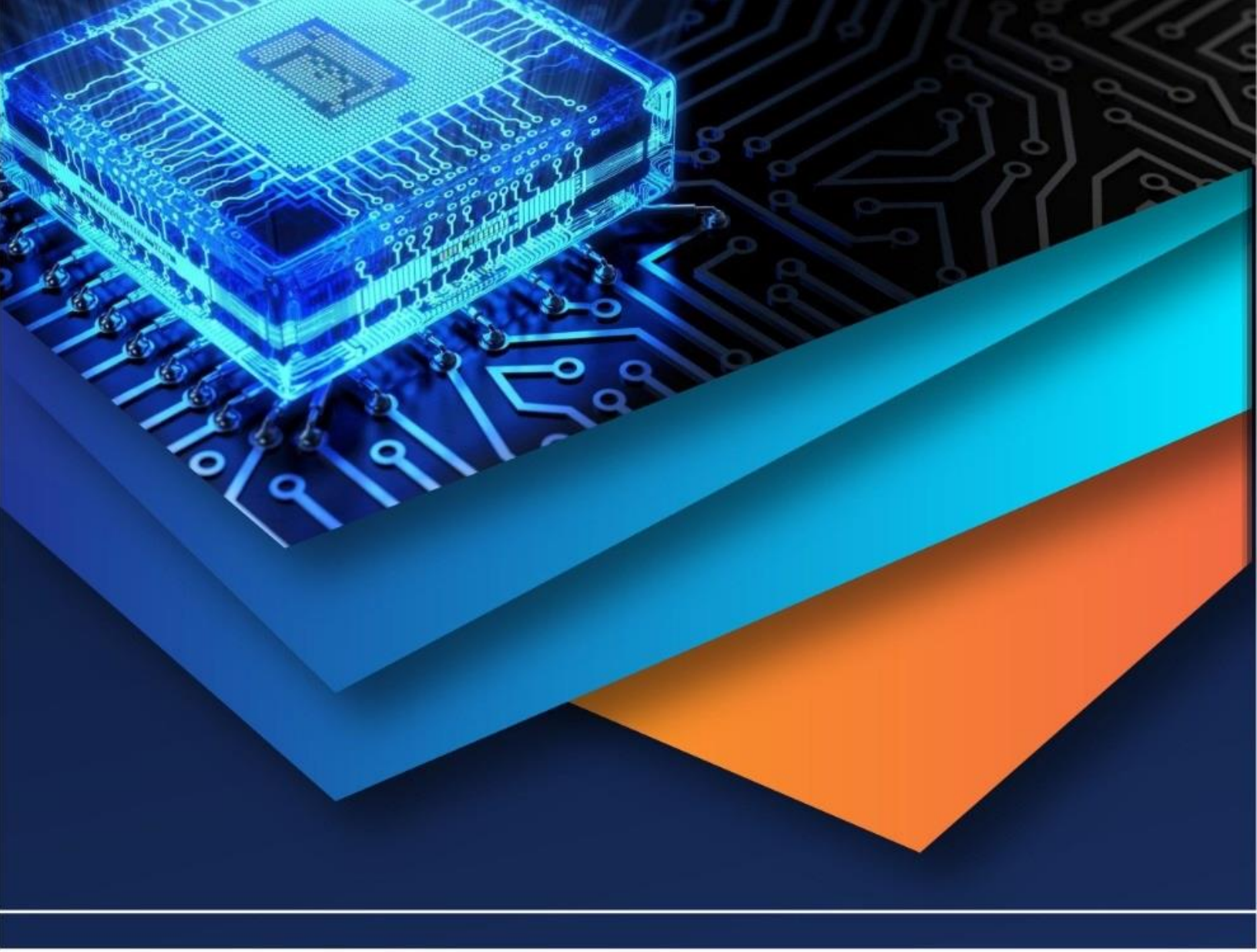
[1] Y. J. Cha, W. Choi, and O. Buyukozturk, "Deep learning-based crack damage detection using convolutional neural networks," Computer-Aided Civil and Infrastructure Engineering, vol. 32, no. 5, pp. 361–378, 2017.

[2] S. Dorafshan, R. J. Thomas, and M. Maguire, "Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete," Construction and Building Materials, vol. 186, pp. 1031–1045, 2018.

[3] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in Proc. International Conference on Learning Representations (ICLR), 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)