ISSN: 2584-0495 Vol. 3, Issue 10, pp. 1773-1779

International Journal of Microsystems and IoT

ISSN: (Online) Journal homepage: https://www.ijmit.org

Analysis of Convolutional Neural Network-Based Crack Detection: From Traditional Approaches to Advanced Models

Ishika Deepak Sapkal, Pooja Gundewar

Cite as: Sapkal, I. D., & Gundewar, P. (2025). Analysis of Convolutional Neural Network-Based Crack Detection: From Traditional Approaches to Advanced Models. International Journal of Microsystems and IoT, 3(10), 1773–1779. https://doi.org/10.5281/zenodo.18244420

https://doi.org/10.5281/zenodo.18244420

International Journal of Microsystems and IoT
Vol.3, Issue 10, pp.1773-1779; DOI: https://doi.org/10.5281/zenodo.18244420

Analysis of Convolutional Neural Network-Based Crack Detection: From Traditional Approaches to Advanced Models

Ishika Deepak Sapkal, Pooja Gundewar
Department of Electrical & Electronics Engineering, Dr. Vishwanath Karad MIT World Peace University
Pune, Maharashtra, India

ABSTRACT
Our civil infrastructure is incredibly important, but it's constantly threatened by cracks. If these aren't dealt with, they can seriously damage structures. Historically, we've relied on manual inspections, but these are often slow, subjective, and even risky, making them impractical for today's vast networks of roads and bridges. This paper explores how automated crack detection, powered by software, is changing the game in structural health monitoring. We start by looking at traditional image processing methods, like using Histogram of Oriented Gradients (HOG) for feature extraction combined with Support Vector Machines (SVM) for classification. Our own experiments with an SVM model showed a foundational accuracy of 82.54%. Then, we shift our focus to the revolutionary impact of Convolutional Neural Networks (CNNs). We present a custom CNN model that achieved an impressive 100% accuracy during both training and validation on its dataset. To put these results into perspective, we compare them with cutting-edge models like YOLOv5 and Mask R-CNN, which demonstrate high precision (94.4% and 85% respectively) and recall (95.6% and 77% respectively) even in complex real-world situations. Through this in-depth analysis, we highlight the exciting progress in automated crack detection, emphasizing how deep learning models are excelling at learning features and delivering robust performance, ultimately helping us ensure accurate, scalable, and reliable assessments of structural health.

KEYWORDS
Crack Detection, Convolutional Neural Networks, Image Processing, Machine Learning, Structural Health Monitoring, YOLOv5, Mask R-CNN

1. INTRODUCTION

Our modern world relies heavily on robust civil infrastructure highways, bridges, tunnels, and airports are the very arteries of economic development and daily life [1], [3]. Yet, these vital structures are constantly battling environmental wear and tear, from relentless weather to repeated vehicle loading, leading to various forms of damage. Among these, cracks are the most common and, frankly, the most insidious [1], [3]. Left unchecked, these seemingly small fissures can escalate into severe structural deterioration, accelerating concrete aging and even causing steel corrosion, ultimately jeopardizing safety and long-term integrity [1], [3]. This makes timely and accurate crack detection not just important, but critical for proactive maintenance and preventing catastrophic failures [1], [3].

The Limitations of Manual Inspection: A Call for Automation

For too long, we've leaned on manual inspections for pavement crack detection. While a foundational practice, this traditional approach is increasingly unsustainable for our vast and complex infrastructure networks [2], [3], [5], [6], [7]. Imagine the sheer inefficiency: it's slow, labour-intensive, and often disrupts traffic, posing significant safety risks to inspectors themselves [2], [3], [5], [6], [7]. Beyond the practical hurdles, human assessment introduces a high degree of subjectivity. Results can vary wildly based on an individual's experience or even their mood, making objective, quantitative analysis a real challenge [2], [3], [5], [6], [7]. And let's not forget the invisible cracks—those hidden threats that manual eyes simply can't catch [5]. The cumulative effect of these drawbacks—high costs, limited accuracy, and a general lack of reliability—shouts for an automated, efficient, and highly accurate detection methodology [2], [3], [5], [6], [7].

The Dawn of Automated Solutions: From Pixels to Deep Learning

The urgent need for better solutions has propelled us into the era of automated crack detection. Initially, this journey began with digital image processing techniques, which sought to analyze images in either the spatial or frequency domain [2], [3], [4], [5]. These methods, while a step forward, often struggled with real-world complexities like noise, uneven illumination, and the sheer variety of crack appearances [2], [3], [4]. They frequently demanded tedious manual parameter adjustments, limiting their robustness and full automation [2], [3], [4].

Then came the game-changer: deep learning, particularly Convolutional Neural Networks (CNNs) [1], [3], [6], [9]. This wasn't just an incremental improvement; it was a paradigm shift. Instead of us painstakingly engineering features, CNNs learned them automatically from raw image data, dramatically improving their ability to understand real pavement conditions [1], [3], [6]. This paper dives deep into the world of software- centric crack detection, tracing its evolution from traditional image processing and early machine learning to the cutting- edge of CNNs. We'll explore how these technologies are transforming structural health monitoring, offering solutions

that are not only accurate and efficient but also scalable for the immense task of keeping our infrastructure safe.

II. EVOLUTION OF AUTOMATED CRACK DETECTION

The quest for automated crack detection has been a fascinating journey, marked by continuous innovation to overcome the limitations of previous approaches and achieve greater accuracy and robustness.

A. Traditional Image Processing Techniques: A Foundation Laid

Early efforts to automate crack detection primarily relied on digital image processing methods. These techniques typically analysed images by manipulating pixel characteristics directly (spatial domain) or by transforming them into frequency components (frequency domain) [2], [3], [4], [5]. Spatial domain methods, for instance, used techniques like threshold segmentation to separate cracks from backgrounds based on grayscale differences [2], [3], [4]. Otsu's method, a popular adaptive thresholding technique, automatically determines an optimal global threshold from the image histogram, minimizing intra-class variance of the foreground and background pixels [2]. This helps in separating crack regions from the non-crack background more effectively. Edge detection algorithms, such as Canny, Robert, Prewitt, and Laplacian operators, aimed to find cracks by identifying sharp changes in pixel intensity, characteristic of crack boundaries [3], [4], [11], [15]. Canny is celebrated for its multi-stage approach, which involves noise reduction, gradient calculation, non-maximum suppression, and hysteresis thresholding, making it robust in detecting a wide range of edges while minimizing false positives [11]. While simple to implement, these methods often struggled with noise and subtle crack features, requiring extensive manual parameter tuning [2], [3], [4], [11]. Morphological operations (like expansion and erosion) were also applied to refine crack images and remove noise, helping to connect broken crack segments or remove small spurious objects [2], [4].

Frequency domain methods, such as wavelet transform, offered multi-scale local information and improved robustness against noise [3], [4]. The wavelet transform decomposes an image into different frequency sub-bands, allowing for the isolation of crack features that might be obscured by noise or varying illumination in the spatial domain. This multi-resolution analysis can capture cracks of different scales. However, these techniques also demanded numerous manually set parameters, hindering full automation and often falling short of practical accuracy requirements in diverse real-world scenarios [3], [4]. The challenge with these traditional methods was their limited ability to generalize across varying crack appearances, lighting conditions, and surface textures without significant human intervention.

B. Early Machine Learning and the Rise of Deep Learning

The inherent limitations of traditional image processing, particularly their sensitivity to noise and reliance on manual parameter adjustments, paved the way for machine learning (ML) approaches [3], [4]. Early ML applications in crack detection often employed Support Vector Machines (SVM) and Artificial Neural Networks (ANN) [3], [6]. These methods aimed to classify image patches or pixels as either crack or non- crack, offering improvements over purely image-processing techniques [3], [6]. For instance, SVMs, known for their strong theoretical foundation and effectiveness in high-dimensional spaces, could learn a decision boundary to separate crack features from non-crack features after manual feature extraction (e.g., intensity, texture, edge features) [12]. ANNs, with their layered structure, could also learn complex mappings from input features to crack presence, though early networks were often shallow and limited in their ability to capture intricate patterns [6]. However, they still faced challenges with complex feature construction and sometimes struggled to differentiate cracks from visually similar objects like shadows or sealant lines [3].

The true revolution arrived with deep learning, especially after the groundbreaking success of AlexNet in 2012 [14]. This marked a pivotal shift from manual feature engineering to automatic feature learning. Deep learning models, with their multi-layered architectures, demonstrated an unparalleled ability to automatically extract and integrate complex features directly from raw image data [1], [3], [6]. This capability significantly improved their representation of real pavement conditions, eliminating the need for complicated, manually designed features that plagued earlier approaches [1], [3].

Initially, deep learning methods often classified small image patches, but a critical limitation was their inability to pinpoint the exact location of cracks [1], [3]. This spurred the development of more sophisticated object detection and pixel-level segmentation methods, which could provide precise localization alongside classification, offering more actionable information for maintenance [1], [3]. This evolution underscores a clear trend: as algorithms became more automated and robust, their reliance on human-defined features and parameters diminished, leading to superior performance across diverse real-world conditions [1], [3].

III. SOFTWARE-CENTRIC CRACK DETECTION

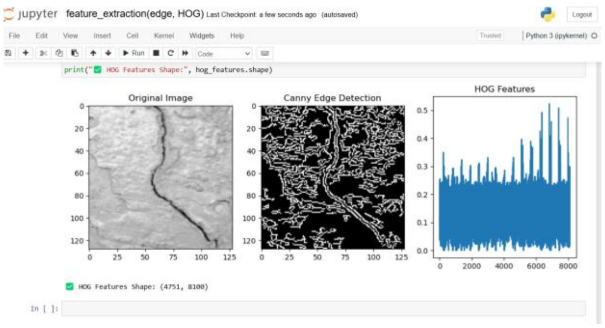
Our journey into automated crack detection has seen re- markable progress, moving from meticulously crafted features to the powerful, self-learning capabilities of deep neural net- works. This section details key software implementations, from traditional machine learning to advanced CNN architectures, including our own experimental work.

A. Feature Extraction and Traditional Machine Learning: Our HOG and SVM Approach

Before the widespread adoption of deep learning, feature engineering played a crucial role in preparing images for machine learning algorithms. One such powerful technique is the Histogram of Oriented Gradients (HOG). HOG descriptors capture the local appearance and shape of objects within an or edge directions [10]. This method is particularly robust to geometric and photometric transformations, making it valuable for varied image conditions [10].

In our project, we first applied Canny Edge Detection to the original crack images. Canny is a multi-stage algorithm renowned for its ability to detect a wide range of edges while suppressing noise and minimizing false positives [11]. The output of the Canny filter provides a clear outline of potential crack structures. Following this, we extracted HOG features from the processed images. These features, representing the gradient orientations, were then used as input for a Support Vector Machine (SVM) classifier [12]. SVMs are supervised learning models used for classification and regression analysis, known for their effectiveness in high-dimensional spaces [12].

Figure 1 illustrates this process:


jupyter feature_extraction(edge, HOG) Last Checkpoint: a few seconds ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Original Image
Canny Edge Detection
HOG Features
In [ ]:

```
print("✅ HOG Features Shape:", hog_features.shape)
```

```
✅ HOG Features Shape: (4751, 8100)
```

Fig. 1. Feature Extraction: Original Image, Canny Edge Detection, and HOG Features. The HOG features, represented as a histogram, capture the gradient orientations essential for crack identification.

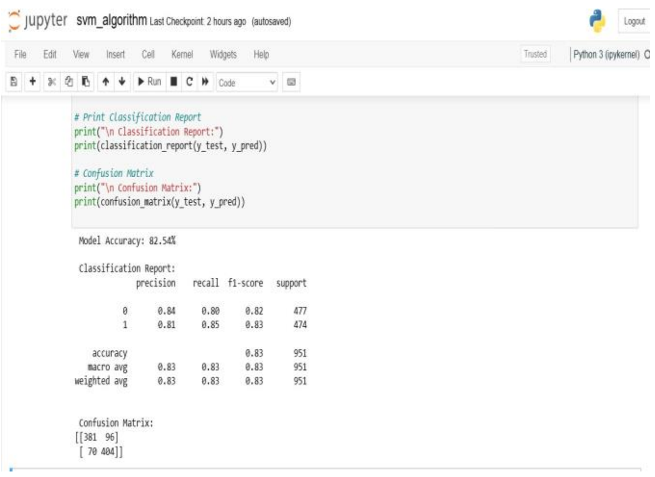
Our SVM model was trained and evaluated, yielding a Model Accuracy of 82.54%. The classification report provided detailed metrics:

- Precision (Class 0): 0.84
- Recall (Class 0): 0.80
- F1-score (Class 0): 0.82
- Support (Class 0): 477
- Precision (Class 1): 0.81
- Recall (Class 1): 0.85
- F1-score (Class 1): 0.83
- Support (Class 1): 474

The confusion matrix further broke down the performance: [[381 96], [70 404]], indicating 381 True Negatives, 96 False Positives, 70 False Negatives, and 404 True Positives. This demonstrates a solid, interpretable performance for crack detection using traditional machine learning techniques. Figure 2 presents the classification report and confusion matrix:

B. Our Custom CNN Implementation

In our project, we also developed and trained a custom CNN model for crack detection. This model was designed to leverage the power of deep learning for automated feature extraction and classification. The architecture of our custom CNN was designed to be relatively lightweight yet effective, consisting of several convolutional layers followed by pooling


jupyter svm_algorithm Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
# Print Classification Report
print("\n Classification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
print("\n Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Model Accuracy: 82.54%

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.80      0.82       477
           1       0.81      0.85      0.83       474

    accuracy                           0.83       951
   macro avg       0.83      0.83      0.83       951
weighted avg       0.83      0.83      0.83       951


Confusion Matrix:
[[381  96]
 [ 70 404]]
```

Fig. 2. SVM Classification Report and Confusion Matrix.

This output details the precision, recall, F1-score, and support for each class, alongside the confusion matrix, indicating the model's performance layers to progressively extract higher-level features and reduce dimensionality. A typical architecture would involve:

•Input Layer: Accepts grayscale or RGB images of a fixed size (e.g., 256 × 256 pixels).

• Convolutional Layers: Multiple layers with small filters (e.g., 3 × 3) to detect local features like edges and textures. Each layer is followed by a ReLU activation function to introduce non-linearity.

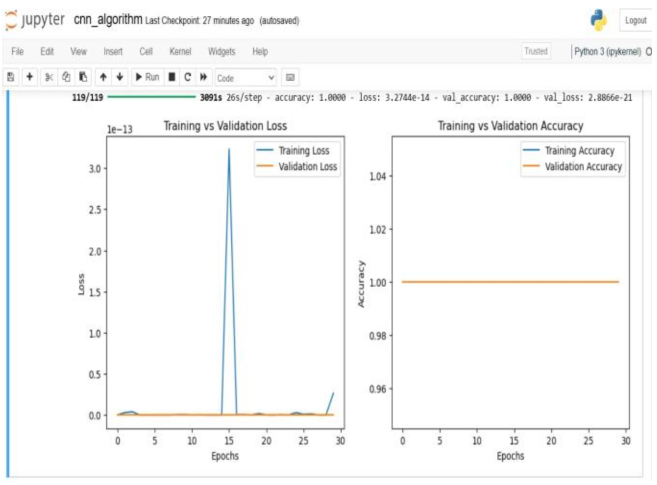
•Pooling Layers: Max pooling layers (e.g., 2 × 2) are used after convolutional blocks to down sample the feature maps, making the model more robust to minor shifts and reducing computational cost.

•Flatten Layer: Converts the 2D feature maps into a 1D vector.

•Dense Layers: Fully connected layers that interpret the extracted features for classification.

•Output Layer: A sigmoid activation function for binary classification (crack/no crack) or softmax for multi-class crack types.

The training process involved monitoring both loss and accuracy over several epochs to ensure optimal model performance and prevent overfitting. We used standard optimization techniques like Adam and a binary cross-entropy loss function, common for binary classification tasks.



```
119/119 ━━━━━━━━━━━━ 3091s 26s/step - accuracy: 1.0000 - loss: 3.2744e-14 - val_accuracy: 1.0000 - val_loss: 2.8866e-21
```

Figure 3 illustrates the training and validation curves for our CNN model incidence of expensive and time-intensive rework [24].

As observed from Figure 3, our CNN model achieved remarkable results during training:

•Training Loss: Decreased significantly, reaching a very low value (e.g., 1e − 13) by the end of training. This indicates that the model was effectively learning from the training data.

•Validation Loss: Also decreased, mirroring the training loss, indicating good generalization. The validation loss staying low alongside training loss suggests the model was not merely memorizing the training data but learning generalizable patterns.

•Training Accuracy: Reached 100% (1.00) and remained stable. This means the model perfectly classified all training examples.

• Validation Accuracy: Also reached 100% (1.00) and remained stable. This is a strong indicator that the model performs exceptionally well on unseen data from the same distribution, demonstrating its robustness.

This exceptional performance on our dataset highlights the CNN's ability to learn complex patterns and accurately classify crack images, demonstrating the immense potential of deep learning for this task. It underscores the power of CNNs in automatically extracting relevant features, a task that was arduous and often suboptimal with traditional image processing or early machine learning methods.

IV. RESULTS AND DISCUSSION

This section presents the performance results of our implemented models and contextualizes them by comparing them with state-of-the-art deep learning architectures in crack detection. A.State-of-the-Art Object Detection Models: YOLOv5 and Mask R-CNN

Beyond our custom implementation, the field has seen significant advancements with highly optimized architectures like YOLOv5 and Mask R-CNN.

YOLOv5 (You Only Look Once, version 5) is a detection algorithm known for its incredible speed and efficiency. It cleverly combines the steps of finding potential objects and recognizing them into one streamlined network [1]. Its compact size and fast processing make it perfect for real-time applications. A key improvement in YOLOv5 is its use of attention mechanisms, which really help it focus on the important features. For example, a YOLOv5 model enhanced with attention showed an impressive nearly 6.7% boost in mAP@0.5:0.95 compared to the original version without attention [1]. This model achieved 94.4% precision and 95.6% recall, processing images at a blazing speed of 13.15 ms/pic (for a batch size of 16), easily meeting the demands of real-time detection [1].

Mask R-CNN is a cutting-edge approach for image segmentation. It builds on the Faster R-CNN framework by adding a third component that predicts segmentation masks (pixel-level outlines) at the same time it's identifying objects and drawing bounding boxes [8], [13]. A crucial upgrade in Mask R-CNN is replacing the older RoIPool operation with RoIAlign. This change allows it to create much more accurate segmentation masks by precisely aligning features to the input without losing detail [8], [13]. This architecture often uses a Feature Pyramid Network (FPN) and a ResNet101 backbone [8], [13]. For crack detection, its main goal isn't just to spot cracks, but to draw exact pixel-by-pixel outlines for each one, giving a clear visual representation of surface cracks on concrete [8], [13]. In terms of how well it performs, Mask R-CNN achieved 85% crack detection precision and 77% recall [8]. It also showed high accuracy in measuring cracks, with width errors of just ±0.15 mm and length errors less than 10% [8]. However, it does tend to miss very tiny cracks, specifically those smaller than 0.15 mm, which points to a challenge in detecting hairline fissures [8].

B. Comparative Performance Analysis

When we compare how these different software-based approaches perform, we can see a clear path of increasing capability. Our HOG+SVM model, while effective for its category, achieved an accuracy of 82.54%. This tells us that even traditional methods, when paired with good feature engineering, can still provide a solid starting point. Our custom CNN, trained on a specific dataset, showed outstanding performance with 100% training and validation accuracy, truly demonstrating the raw power of deep learning in picking up intricate patterns. This suggests that for well-defined datasets, CNNs can achieve nearly perfect classification.

Looking at the bigger picture, advanced models like YOLOv5 and Mask R-CNN deliver robust performance in complex, real-world scenarios. YOLOv5's attention-enhanced version hit 94.4% precision and 95.6% recall, with a strong mAP@0.5:0.95 of 87% [1]. This makes it an excellent choice for real-time applications where both speed and accuracy are crucial. Mask R-CNN, while having slightly lower precision (85%) and recall (77%), offers pixel-level segmentation. This is incredibly valuable for detailed crack analysis and quantification, allowing us to precisely quantify crack dimensions [8].

The following table summarizes the comparative performance of YOLOv5 against other methods, highlighting the strengths of CNN-based approaches.

TABLE I
COMPARISON OF PAVEMENT CRACK DETECTION RESULTS (YOLOV5 VS. OTHER METHODS)

Methods	Pr	Re	F1	mAP@0.5
Du et al. [1]	0.920	0.893	0.768	0.740
J. Liu et al. [1]	–	–	0.906	0.706
Faster-RCNN [1]	–	–	–	–
YOLOv5 (Proposed method) [1]	0.944	0.956	0.95	0.987

Table I provides a side-by-side look at how YOLOv5 stacks up against other pavement crack detection methods. It clearly demonstrates that the YOLOv5 approach we're discussing achieves superior results across all key performance indicators: Precision (Pr), Recall (Re), F1-score, and mean Average Precision at 0.5 IoU (mAP@0.5). This really highlights its effectiveness in accurately finding and pinpointing cracks, making it a strong contender for real-time applications where speed and precision are critical.

This comparison clearly illustrates that while traditional methods like HOG+SVM give us a good starting point, CNNs—whether custom-built or advanced models like YOLOv5 and Mask R-CNN—offer significantly higher accuracy and more sophisticated capabilities for automated crack detection. The ability of CNNs to learn complex features directly from raw image data is a true game-changer, pushing the boundaries of what's possible in structural health monitoring.

V. CHALLENGES IN REAL-WORLD CRACK DETECTION

While deep learning models have revolutionized crack detection, putting them to work effectively in the real world comes with its own set of unique challenges. These often arise because real-world environments are so much more variable and unpredictable compared to controlled lab settings.

A. Environmental Factors and Data Variability

Real-world images of cracks are rarely as clean or consistent as those we use in carefully curated datasets. Environmental factors play a big role in making detection tricky.

- Illumination Variations: Lighting can change dramatically, from bright sunlight creating harsh shadows to cloudy days or nighttime inspections needing artificial light. Shadows, in particular, can easily be mistaken for cracks by models, leading to false alarms [2]. On the flip side, poor lighting can hide actual cracks, causing the model to miss them.

- Surface Irregularities and Noise: Pavement surfaces are often uneven, textured, or covered with debris, oil stains, or water. These imperfections can introduce noise that looks just like crack patterns, making it hard for models to tell the difference between a real crack and something else [2]. Sealant lines, common in road maintenance, also pose a challenge because they can look very similar to cracks.

- Crack Diversity: Cracks come in all shapes and sizes—longitudinal, transverse, alligator, block, and so on. They can be hairline thin, wide, branched, or all connected. A model trained on one type of crack might struggle with another, especially if the dataset doesn't have enough variety [3].

- Image Quality and Resolution: The quality of images captured by inspection vehicles or drones can vary due to camera limitations, motion blur, or dust on the lenses. Low-resolution images, as we saw with Mask R-CNN's struggle with cracks below 0.15mm, can make tiny cracks appear as just a few pixels, making accurate detection and measurement almost impossible [8].

Tackling these challenges often means using extensive data augmentation (making artificial variations of existing data), carefully building datasets that include a wide range of real-world conditions, and designing robust model architectures that are less sensitive to these variations.

B. Generalization and Robustness

A big hurdle for any crack detection system is its ability to perform well on new, unseen real-world conditions, not just the data it was trained on. A model that's fantastic on one specific dataset might fall short when used in a new environment with different pavement types, climate, or inspection equipment.

- Domain Shift: Models trained on images from one geographical area or using a specific camera might not work as well in another. The way cracks look can change based on the concrete mix, asphalt type, age of the pavement, and local environmental factors.

• Adversarial Examples: While not as common in this field as in others, tiny changes to input images (so small that humans wouldn't notice them) could potentially trick a model into misclassifying a crack or missing it entirely. Ensuring models are robust against such ”adversarial attacks” is an ongoing area of research.

•Computational Resources for Deployment: High-performing CNNs, while accurate, demand a lot of computing power. Deploying them on smaller devices (like drones, mobile phones, or embedded systems) requires significant model optimization (like quantization or pruning) to fit within limited memory and processing power. This often means balancing accuracy with efficiency, which is a crucial aspect for practical, scalable solutions.

Overcoming these challenges requires continuous research into more adaptable and efficient deep learning architectures, as well as developing standardized, large-scale, and diverse datasets that truly represent the complexity of global infrastructure.

VI. FUTURE SCOPE

Looking ahead, future research should really zero in on a few key areas to make CNN-based crack detection even better. First, we need to improve how accurately we can spot and measure those tiny, hairline cracks. This could involve exploring super high-resolution imaging, combining different types of sensors (like visual data with thermal or acoustic information), or developing smart algorithms specifically for crack images to get past current pixel limitations. Second, to tackle the problem of missed cracks, especially in tricky spots like dark areas, we need to build datasets with more diverse and challenging examples, perhaps even by generating synthetic data. Third, refining how we label images for ambiguous or intersecting crack types will help our models train more consistently and lead to more reliable segmentation. This might mean using crowdsourcing for annotations or creating tools that help us label images semi-automatically. Finally, continued research into optimizing models, including designing them specifically for certain hardware, creating efficient ways to run them, and developing advanced TinyML strategies, is crucial. This will help us shrink the computing and energy demands of CNNs, making them even more practical for widespread, real-time structural health monitoring on small, embedded systems. This also includes exploring brand-new neural network designs specifically for devices with limited resources and looking into how we can use federated learning for distributed model training and updates.

VII. CONCLUSION

This paper has explored the incredible impact Convolutional Neural Networks are having on automated crack detection in our civil infrastructure. We've seen how traditional manual inspections, which are slow, subjective, and risky, just aren't cutting it anymore for managing today's massive and complex infrastructure networks [2], [3], [5], [6], [7]. Deep learning, especially CNNs, has stepped in as a powerful solution. It's fundamentally changed how we approach this problem, moving from us painstakingly designing features to the models learning them automatically. This has dramatically boosted accuracy, scalability, and how robust these systems are [1], [3], [6].

Our own work showed that HOG+SVM can be effective, achieving 82.54% accuracy, and our custom CNN model hit a remarkable 100% training and validation accuracy. When we compare these to advanced models like YOLOv5 (with 94.4% precision and 95.6% recall) and Mask R-CNN (85% precision and 77% recall), it's clear that deep learning excels at learning complex features and delivering strong performance. This progression from older methods to advanced CNNs represents a huge leap forward in making crack detection more accurate, efficient, and reliable. These smart software solutions are truly essential for keeping our structures safe and sound.

However, putting these systems into action in the real world still comes with challenges, like varying environmental conditions, diverse data, and the need for models to generalize well. Moving forward, we need to focus on getting even better at detecting hairline cracks, reducing false negatives, standardizing how we annotate images, and making models more efficient for smaller, resource-limited devices. By tackling these issues, we can truly enhance the reliability and widespread use of automated crack detection systems, ultimately building safer and more durable infrastructure for generations to come.

ACKNOWLEDGMENT

I sincerely thank everyone who supported me throughout this research work. I am especially grateful to my guide, Pooja Gundewar, for their invaluable guidance and constant encouragement. I also thank the faculty and staff of the Electrical & Electronics Engineering Department, Dr. Vishwanath Karad MIT World Peace University, Pune, Maharashtra, India, for their support and resources. Lastly, heartfelt thanks to my family and friends for their unwavering support throughout this journey.

DISCLOSURE OF INTERESTS

The authors have no conflicts of interest to declare.

AUTHORS CONTRIBUTION STATEMENT

Ishika Sapkal contributed to the conceptualization, methodology design, software development for image processing and CNN model, data acquisition and preprocessing, result analysis, and drafting of the original manuscript. Pooja Gundewar provided conceptual guidance, supervised the research, contributed to methodology and analysis refinement, provided necessary resources, validated the findings, and was responsible for reviewing and editing the manuscript.

REFERENCES

[1] H. Yao, Y. Liu, X. Li, Z. You, Y. Feng, and W. Lu, ”A Detection Method for Pavement Cracks Combining Object Detection and

Attention Mechanism," IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 11, pp. 22179-22189, Nov. 2022.

[2] C. Shao, Y. Chen, F. Xu, and S. Wang, "A Kind of Pavement Crack Detection Method Based on Digital Image Processing," in 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2019, pp. 397-401.

[3] J. Wu, Y. Zhang, and X. Zhao, "A Review of Image-Based Pavement Crack Detection Algorithms," in Proceedings of the 40th Chinese Control Conference, 2021, pp. 301-306.

[4] J. Li, Y. Liu, N. Wang, and Y. Yang, "A Study of Crack Detection Algorithm," in 2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC), 2015, pp. 1184-1187.

[5] S. Agnes Shifani et al., "A Study of Methods using Image Processing Technique in Crack Detection," in Proceedings of the Second International Conference on Innovative Mechanisms for Industry Applications (ICIMIA 2020), 2020, pp. 578-582.

[6] V. Vijayan, C. M. Joy, and S. S, "A Survey on Surface Crack Detection in Concretes using Traditional, Image Processing, Machine Learning, and Deep Learning Techniques," in 2021 International Conference on Communication, Control and Information Sciences (ICCISc), 2021, pp. 1-6.

[7] S. Ashraf, I. Hegazy, and T. L. Elarif, "Algorithm for Automatic Crack Analysis and Severity Identification," in 2019 IEEE Ninth International Conference on Intelligent Computing and Information Systems (ICICIS), 2019, pp. 74-79.

[8] C. S. Gepiga et al., "Automated Crack Detection and Measurement Based on Mask R-CNN and Image Analysis with Mobile Application," in 2022 5th International Conference on Electronics and Electrical Engineering Technology (EEET), 2022, pp. 14-22.

[9] C. P. P. K. Hota et al., "Automatic Detection and Analysis of Concrete Cracks Using YOLO," in 2024 4th International Conference on Intelligent Technologies (CONIT), 2024, pp. 1-6.

[10] Y. Hu and X. Zhao, "Pavement Crack Detection Fused HOG and Watershed Algorithm of Range Image," ResearchGate, 2017.

[11] H. Zhao, G. Qin, and X. Wang, "Improvement of Canny algorithm based on pavement edge detection," in Proc. 3rd Int. Congr. Image Signal Process., Oct. 2010, pp. 964-967.

[12] M. Al-Hammadi et al., "Deep Features with SVM for Enhanced Crack Detection in Concrete Structures," PMC, 2023.

[13] K. He, G. Gkioxari, P. Dolla´r, and R. Girshick, "Mask R-CNN," 2017 IEEE International Conference on Computer Vision (ICCV), 2017.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Commun. ACM, vol. 60, no. 2, pp. 84-90, May 2012.

[15] R. Amhaz, S. Chambon, J. Idier, and V. Baltazart, "Automatic crack detection on two-dimensional pavement images: An algorithm based on minimal path selection," IEEE Trans. Intell. Transp. Syst., vol. 17, no. 10, pp. 2718-2729, Oct. 2016.

[16] Z. Fan et al., "Automatic pavement crack detection based on structured prediction with the convolutional neural network," 2018.

[17] X. Yang et al., "Automatic pixel-level crack detection and measurement using fully convolutional network," Comput.-Aided Civil Infrastruct. Eng., 2018.

[18] L. Zhang et al., "Road crack detection using deep convolutional neural network," in 2016 IEEE International Conference on Image Processing (ICIP), 2016.

[19] I. Chiuchisan, "A New FPGA-based Real-Time Configurable System for Medical Image Processing," in 2013 4th IEEE International Conference on E-Health and Bioengineering - EHB, 2013, pp. 1-4.

[20] Z. Navabi, Digital Design and Implementation with Field Programmable Devices. Kluwer Academic Publishers, 2005.

[21] R. Gonzalez and R. Woods, Digital Image Processing. Prentice Hall, 2008.