

Comparative Analysis of Conventional Machine Learning Classifiers for Concrete Crack Detection Using PCA-Based Pixel Features

Anindita Kundu Mondal¹ and Dr. Aruna Chamatkar²

Research Scholar, Department of Electronics and Computer Science, RTMNU, Nagpur, India¹

Associate Professor, MCA Department, Kamala Nehru Mahavidyalaya, Nagpur, Maharashtra, India²

Abstract: Concrete crack detection is essential for continuous structural inspection and maintenance. This study presents a comparative evaluation of conventional machine-learning classifiers for concrete crack classification using the NYA-Crack-Data dataset containing 5,026 images, including 2,167 crack and 2,859 no-crack images. Images were converted to grayscale and resized to 64×64 pixels, generating 4,096 pixel-level features. Incremental Principal Component Analysis (PCA) was used for dimensionality reduction, and 300 and 500 components were evaluated. Although 500 components retained 96.02% variance, the 300-component representation achieved better performance, retaining 93.64% variance with 79.11% accuracy, 76.26% F1-score, 78.95% balanced accuracy, and 86.43% ROC-AUC. Therefore, 300 components were selected for final classification. Five classifiers: Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbours (KNN), Logistic Regression (LR), and Decision Tree (DT), were evaluated using the same PCA-reduced features. On the independent test set, SVM achieved the highest accuracy (78.91%), sensitivity (82.46%), F1-score (77.12%), and balanced accuracy (79.34%), while Random Forest achieved the highest precision (76.36%), specificity (82.75%), and ROC-AUC (87.54%). Five-fold cross-validation further indicated stable performance. The findings demonstrate that conventional ML methods provide a useful baseline for concrete crack classification using PCA-reduced pixel features.

Keywords: Concrete Crack Detection; Machine Learning; PCA; Support Vector Machine; Random Forest.

I. INTRODUCTION

Surface cracks can indicate deterioration and may require further assessment of their severity and structural implications. Conventional visual inspection and image-processing techniques can be affected by illumination, surface texture, noise, shadows, and other visual disturbances [1]–[4]. Therefore, automated image-based crack detection has become an important research area for efficient and repeatable infrastructure inspection.

Early automated methods primarily relied on filtering, edge detection, thresholding, and morphological operations. Chen and Hutchinson proposed an image-based framework for concrete crack monitoring and quantification [1], while Nishikawa et al. investigated sequential image filtering for crack detection and width estimation [2]. Abdel-Qader et al. evaluated edge-detection techniques for bridge crack identification [3], and Mohan and Poobal reviewed image-processing approaches and their associated challenges [4]. Although these methods established important foundations, their performance can depend on manually selected parameters and engineered image characteristics.

Conventional machine-learning approaches subsequently provided an alternative by combining image representations with supervised classifiers. Sparse representation and feature-based methods have been investigated for concrete crack recognition [5]–[7]. Wang et al. employed L2 sparse representation [5], Chun et al. applied Light Gradient Boosting Machine [6], and Wang et al. developed an SVM-based crack recognition system [7]. These studies demonstrate the



potential of conventional machine learning; however, comparisons are often coupled with different feature-engineering strategies, making direct comparison among classifiers difficult.

Deep learning has since become dominant in automated crack detection because of its ability to learn visual representations directly from images. CNN-based methods have demonstrated effective crack detection and classification [8]–[10], while the SDNET2018 dataset further supported systematic evaluation of concrete crack detection methods [11]. Subsequent studies have explored hierarchical feature learning, semantic segmentation, transfer learning, object detection, crack measurement, and fully convolutional architectures [12]–[23]. More recent research has investigated residual networks, multiresolution analysis, crack quantification, robustness to illumination and shadows, vision transformers, and three-dimensional reconstruction [24]–[33]. In the broader context of non-destructive evaluation, Kundu Mondal and Chamatkar reviewed the evolution of AI methodologies for cultural heritage structures, highlighting the progression from conventional machine learning and handcrafted representations toward deep-learning-based approaches [17].

This study presents a systematic comparative analysis of five conventional machine-learning classifiers: Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbors (KNN), Logistic Regression (LR), and Decision Tree (DT): for binary concrete crack classification using the NYA-Crack-Data dataset. Images are converted to grayscale, resized to 64×64×64 pixels, and represented as 4,096-dimensional pixel vectors. Incremental Principal Component Analysis (PCA) is used to obtain a compact representation, with 300 and 500 components evaluated. The classifiers are assessed using accuracy, precision, sensitivity, specificity, F1-score, balanced accuracy, Matthews correlation coefficient (MCC), and ROC-AUC. Five-fold cross-validation and targeted hyperparameter optimization are also performed to assess performance stability. The study consequently provides a reproducible conventional-machine-learning benchmark for concrete crack classification and complements the predominantly deep-learning-oriented literature.

II. LITERATURE REVIEW

Automated concrete crack detection has progressed from conventional image processing to machine-learning and deep-learning approaches. Early studies focused on filtering, edge detection, crack monitoring, and quantification. Chen and hutchinson [1], nishikawa et al. [2], and abdel-qader et al. [3] investigated image-based crack monitoring, sequential filtering, and edge-detection techniques, respectively. Mohan and poobal [4] reviewed these approaches and highlighted challenges related to noise, segmentation, feature extraction, and imaging conditions. Although effective for specific applications, these methods depend strongly on manually designed processing procedures.

Conventional machine-learning methods subsequently introduced supervised classification using image-derived representations. Wang et al. proposed L2 sparse representation for fast crack detection [5], while Chun et al. applied LightGBM [6] and Wang et al. developed an SVM-based crack recognition system [7]. These studies demonstrate the potential of conventional classifiers, although their performance is often associated with specific engineered or handcrafted features. This motivates controlled comparisons in which different classifiers operate on a common feature representation.

The emergence of CNNs enabled automatic feature learning directly from images. Cha et al. [8] and Kim and Cho [9] demonstrated CNN-based concrete crack detection, while Dorafshan et al. compared deep CNNs with traditional edge detectors [10]. The SDNET2018 dataset introduced by Dorafshan et al. [11] further supported systematic evaluation of concrete crack detection. Subsequent research expanded toward hierarchical feature learning, fully convolutional networks, bridge crack detection, and other deep architectures [12]–[16]. In the broader context of non-destructive evaluation, Kundu Mondal and Chamatkar [17] reviewed the progression from classical machine learning and handcrafted features to CNN-, GAN-, and attention-based approaches for cultural-heritage assessment.

Deep-learning research has further addressed semantic segmentation, transfer learning, crack measurement, object detection, and tunnel inspection [18]–[23]. More recent studies have investigated multiresolution analysis, residual networks, crack quantification, pixel-level detection, illumination and shadow robustness, vision transformers, and



three-dimensional reconstruction [24]–[31]. These developments demonstrate the increasing emphasis on robust and context-aware automated inspection. However, deep-learning approaches can require greater computational resources and training data, maintaining the relevance of conventional machine-learning models as efficient and reproducible baselines.

Recent reviews continue to identify challenges related to feature representation, robustness, computational efficiency, and practical deployment [32], while lightweight architectures such as Mobile-DenseNet have been proposed for efficient crack detection [33]. Overall, the literature demonstrates a clear progression from handcrafted image processing to conventional machine learning and subsequently to CNNs, segmentation networks, object detectors, transformers, and hybrid architectures.

Despite the strong performance of deep-learning methods, a controlled evaluation of conventional classifiers remains useful for establishing reproducible baselines. Therefore, this study compares SVM, Random Forest, KNN, Logistic Regression, and Decision Tree using a common PCA-reduced grayscale pixel representation. Incremental PCA is used for dimensionality reduction, with 300 and 500 components evaluated. The classifiers are compared using accuracy, precision, sensitivity, specificity, F1-score, balanced accuracy, MCC, and ROC-AUC. This framework examines both classifier performance and the effect of PCA dimensionality, providing a controlled conventional-machine-learning baseline for concrete crack classification.

III. METHODOLOGY

A. Proposed Framework

The proposed framework provides a controlled comparison of conventional machine-learning classifiers for binary concrete crack classification. The workflow consists of dataset preparation, grayscale conversion and resizing, pixel-feature extraction, Incremental Principal Component Analysis (IPCA), classification, and performance evaluation. The same PCA-reduced representation is supplied to Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbors (KNN), Logistic Regression (LR), and Decision Tree (DT).

B. Dataset and Preprocessing

The NYA-Crack-Data dataset [8] contains 5,026 concrete image patches of size 227×227 pixels, comprising 2,167 crack and 2,859 no-crack images. A stratified image-level split produced 3,518 training, 754 validation, and 754 testing images. RGB images were converted to grayscale and resized to 64×64 pixels. Each image was flattened into a 4,096-dimensional pixel vector.

C. PCA-Based Dimensionality Reduction

IPCA was fitted using training data only and then applied to the validation and test sets. Representations with 300 and 500 components were evaluated. The 300-component representation retained 93.64% variance and produced the best observed performance; therefore, 300 components were selected for the final comparison.

D. Classification

SVM, RF, KNN, LR, and DT were trained using the common 300-component representation. Hyperparameter optimization for SVM and RF used five-fold cross-validation with F1-score as the selection criterion. The best SVM used an RBF kernel with $C = 1$ and $\gamma = 0.1$. The best RF used 200 trees, maximum depth 30, `max_features = sqrt`, minimum samples split = 5, and minimum samples leaf = 1. KNN used five neighbors; LR used `max_iter = 2000` with balanced class weights; and DT used balanced class weights.

E. Evaluation and Validation

The models were evaluated using accuracy, precision, sensitivity (recall), specificity, F1-score, balanced accuracy, Matthews correlation coefficient (MCC), and ROC-AUC. Five-fold cross-validation was used to assess performance stability, with mean and standard deviation reported for the principal metrics. The independent test set was used for final comparison.



TABLE I: DATASET DISTRIBUTION

Subset	Total	Crack	No Crack
Training	3,518	1,517	2,001
Validation	754	325	429
Testing	754	325	429
Total	5,026	2,167	2,859

TABLE II: FEATURE DIMENSIONALITY

Dataset	Original Features	PCA Features
Training	4,096	300
Validation	4,096	300
Testing	4,096	300

IV. EXPERIMENTAL SETUP

A. Computational Environment

All experiments were implemented in a Google Colab environment using a CPU-based runtime. The computational environment used Python 3.13.15, NumPy 2.1.3, and scikit-learn 1.6.1. A fixed random state of 42 was used wherever a random-state parameter was specified to improve reproducibility.

B. Dataset and Data Partitioning

The experiments used the NYA-Crack-Data dataset [8], containing 5,026 image patches divided into 2,167 crack and 2,859 no-crack images. The dataset was partitioned at the image level into training, validation, and testing subsets containing 3,518, 754, and 754 images, respectively. Stratification was used to preserve the class distribution across the subsets. The training set was used for model development, the validation set for comparative model assessment, and the testing set for final performance evaluation.

C. Image Preprocessing and Feature Construction

Each input image was converted from RGB to grayscale and resized to 64×64 pixels. The resulting grayscale image was flattened to form a 4,096-dimensional pixel vector. Accordingly, the training, validation, and testing feature matrices contained $3,518 \times 4,096$, $754 \times 4,096$, and $754 \times 4,096$ elements, respectively. The same preprocessing and pixel representation were used for all five classifiers.

D. PCA-Based Dimensionality Reduction

Incremental Principal Component Analysis (IPCA) was employed to reduce the dimensionality of the 4,096-dimensional pixel representation. The IPCA transformation was fitted using the training data only and subsequently applied to the validation and testing sets. Two successfully evaluated representations, containing 300 and 500 components, were compared. The 300-component representation retained 93.64% of the variance and achieved 79.11% accuracy, 76.26% F1-score, 78.95% balanced accuracy, and 86.43% ROC-AUC in the corresponding cross-validation evaluation. The 500-component representation retained 96.02% variance but achieved slightly lower performance. Therefore, 300 components were selected for the final classifier comparison.

E. Classifier Configuration

Five conventional machine-learning classifiers were evaluated using the same PCA-reduced feature representation. SVM used an RBF kernel with $C = 1$ and $\gamma = 0.1$. Random Forest used 200 estimators, maximum depth of 30, $\max_features = \text{sqrt}$, minimum samples split of 5, and minimum samples leaf of 1. KNN used $n_neighbors = 5$ and $n_jobs = -1$. Logistic Regression used $\max_iter = 2000$, $class_weight = \text{balanced}$, and $random_state = 42$. Decision Tree used $random_state = 42$ and $class_weight = \text{balanced}$.

F. Model Training and Validation Protocol

The classifiers were trained using the PCA-transformed training data. Initial model comparison was performed on the validation set using accuracy, precision, recall, F1-score, and balanced accuracy. The held-out testing set was then used



for final evaluation using accuracy, precision, sensitivity, specificity, F1-score, balanced accuracy, Matthews correlation coefficient (MCC), and ROC-AUC. Confusion matrices and ROC curves were generated to further analyze classification behavior.

G. Five-Fold Cross-Validation

Five-fold cross-validation was conducted to assess model stability. In each iteration, four folds were used for training and the remaining fold for validation, with every fold serving as the validation fold once. Mean performance and standard deviation were calculated for accuracy, precision, recall, F1-score, balanced accuracy, and ROC-AUC. The obtained mean results were $79.11 \pm 1.41\%$ accuracy, $74.76 \pm 1.62\%$ precision, $77.85 \pm 2.15\%$ recall, $76.26 \pm 1.67\%$ F1-score, $78.95 \pm 1.46\%$ balanced accuracy, and $86.43 \pm 1.55\%$ ROC-AUC.

H. Hyperparameter Optimization

Hyperparameter optimization was performed separately for Random Forest and SVM using five-fold cross-validation. For Random Forest, 15 candidate combinations were evaluated, resulting in 75 model fits. The best configuration achieved a five-fold F1-score of 76.09%. For SVM, 16 candidate combinations were evaluated, resulting in 80 model fits. The best configuration achieved a five-fold F1-score of 75.41%.

I. Performance Evaluation Criteria

Accuracy, precision, sensitivity, specificity, F1-score, balanced accuracy, MCC, and ROC-AUC were used to provide complementary measures of classification performance. Training time was also recorded for the classifier experiments. Multiple metrics were considered because the dataset contains more no-crack than crack images.

J. Reproducibility and Experimental Scope

A fixed random state of 42 was used for reproducible operations where applicable. The final framework was restricted to conventional machine-learning classifiers operating on PCA-reduced grayscale pixel features. Handcrafted descriptors, convolutional neural networks, and the attempted RGB-versus-grayscale comparison were not included in the final experimental analysis.

V. EXPERIMENTAL RESULTS AND DISCUSSION

A. Validation Performance

Using the selected 300-component PCA representation, Random Forest achieved the strongest validation performance, with 77.59% accuracy and 73.80% F1-score. KNN showed the weakest crack detection, with 27.69% recall and 39.13% F1-score.

TABLE III VALIDATION PERFORMANCE

Model	Precision (%)	Recall (%)	F1 (%)	Accuracy (%)	Bal. Acc. (%)
Random Forest	74.38	73.23	73.80	77.59	77.06
Decision Tree	67.39	66.77	67.08	71.75	71.15
KNN	66.67	27.69	39.13	62.86	58.60
Logistic Regression	60.00	56.31	58.10	64.99	63.93
SVM	71.12	60.62	65.45	72.41	70.98

B. Independent Test-Set Performance

On the independent test set of 754 images, SVM achieved the highest accuracy (78.91%), sensitivity (82.46%), F1-score (77.12%), balanced accuracy (79.34%), and MCC (0.58). Random Forest achieved the highest precision (76.36%), specificity (82.75%), and ROC-AUC (87.54%). These results indicate a trade-off between crack sensitivity and no-crack specificity.



TABLE IV FINAL TEST-SET PERFORMANCE (%) EXCEPT MCC

Model	Precision	Sensitivity	Specificity	Accuracy	F1	Bal. Acc.	ROC-AUC	MCC
SVM	72.43	82.46	76.22	78.91	77.12	79.34	87.35	0.58
Decision Tree	62.39	62.77	71.33	67.64	62.58	67.05	67.05	0.34
Random Forest	76.36	73.54	82.75	78.78	74.92	78.14	87.54	0.57
Logistic Regression	63.01	61.85	72.49	67.90	62.42	67.17	70.48	0.34
KNN	75.34	33.85	91.61	66.71	46.71	62.73	71.77	0.32

C. Confusion-Matrix and ROC Analysis

The confusion matrices in Fig. 1 provide the class-wise error distribution of the five classifiers. SVM produced the strongest crack detection, whereas KNN showed a strong bias toward the no-crack class, resulting in high specificity but many missed crack samples. The matrices were generated from the final test-set results.

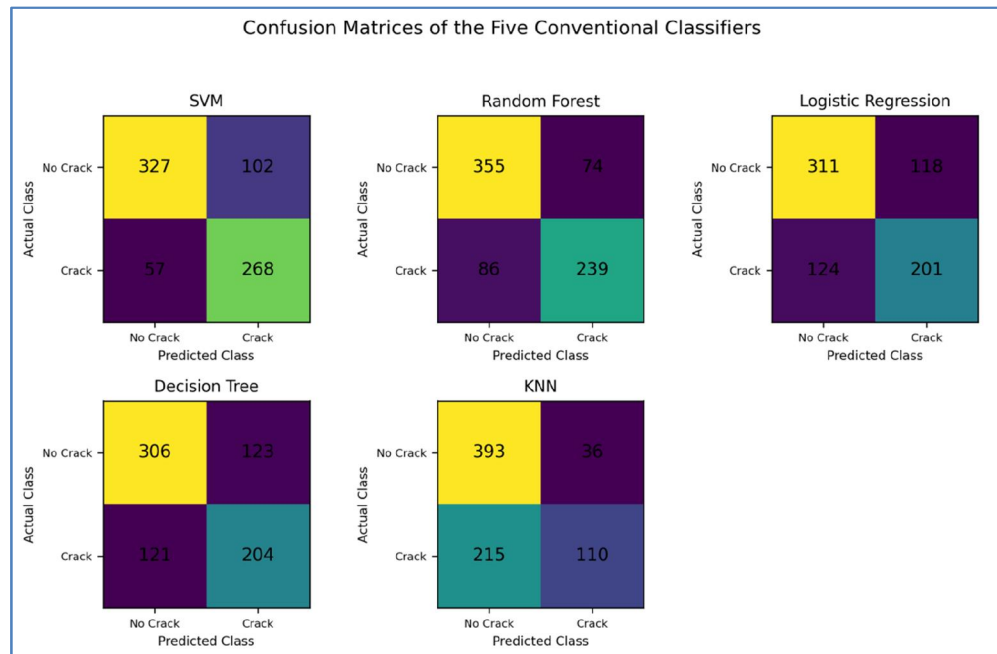


Fig. 1. Confusion matrices of the five conventional classifiers on the independent test set.

ROC analysis showed comparable discrimination for SVM and Random Forest, with ROC-AUC values of 87.35% and 87.54%, respectively. Random Forest achieved the highest ROC-AUC, whereas SVM provided higher sensitivity at the selected operating threshold.



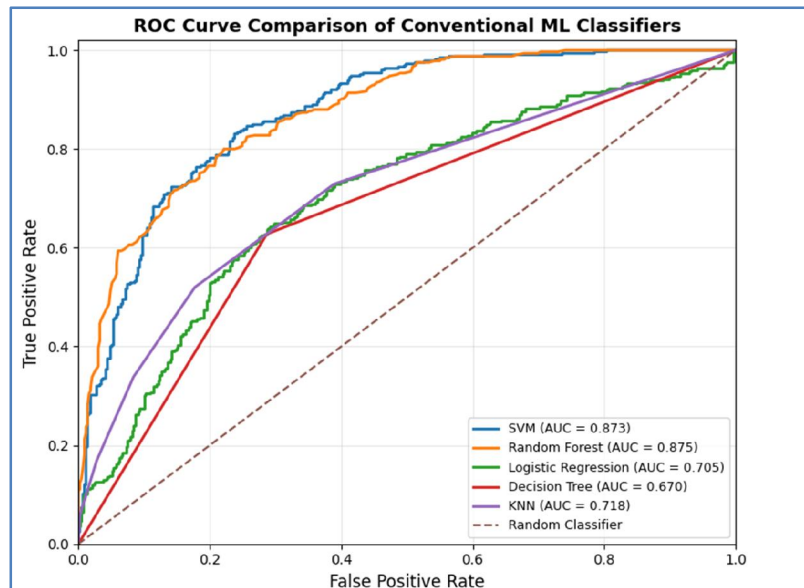


Fig. 2. ROC curves of the evaluated conventional machine-learning classifiers on the independent test set.

D. Cross-Validation and PCA Analysis

Five-fold cross-validation produced $79.11 \pm 1.41\%$ accuracy, $76.26 \pm 1.67\%$ F1-score, and $86.43 \pm 1.55\%$ ROC-AUC, indicating reasonably stable performance across folds.

TABLE V FIVE-FOLD CROSS-VALIDATION RESULTS

Metric	Mean $\pm$ SD
Accuracy	$79.11 \pm 1.41\%$
Precision	$74.76 \pm 1.62\%$
Recall	$77.85 \pm 2.15\%$
F1-score	$76.26 \pm 1.67\%$
Balanced Accuracy	$78.95 \pm 1.46\%$
ROC-AUC	$86.43 \pm 1.55\%$

PCA dimensionality analysis showed that 300 components outperformed 500 components. The 300-component representation retained 93.64% variance and achieved 79.11% accuracy and 86.43% ROC-AUC, compared with 78.82% accuracy and 85.91% ROC-AUC for 500 components. Thus, retaining more variance did not necessarily improve classification performance.

TABLE VI EFFECT OF PCA COMPONENT DIMENSIONALITY

Components	Variance (%)	Accuracy (%)	F1 (%)	Bal. Acc. (%)	ROC-AUC (%)
300	93.64	79.11	76.26	78.95	86.43
500	96.02	78.82	75.75	78.58	85.91

E. Comparative Discussion

Overall, SVM and Random Forest were the strongest classifiers, but their advantages differed. SVM achieved the highest test accuracy and sensitivity, making it preferable when crack detection is prioritized. Random Forest provided higher precision, specificity, and ROC-AUC, indicating stronger no-crack recognition and comparable overall



discrimination. Decision Tree showed intermediate performance, while Logistic Regression and KNN were less effective.

The results also show that the compact 300-component representation was sufficient for effective classification. Overall, PCA-reduced grayscale pixel features combined with conventional machine learning provide a reproducible and computationally efficient baseline for concrete crack classification.

VI. CONCLUSION

This study presented a systematic comparison of five conventional machine-learning classifiers: SVM, Random Forest, Logistic Regression, Decision Tree, and KNN, for binary concrete crack classification using the NYA-Crack-Data dataset. All classifiers employed a common PCA-reduced grayscale pixel representation to ensure a controlled comparison. The 300-component PCA representation retained 93.64% variance and achieved 79.11% accuracy, 76.26% F1-score, 78.95% balanced accuracy, and 86.43% ROC-AUC, outperforming the 500-component representation despite its lower variance retention. On the test set, SVM achieved the highest accuracy (78.91%) and sensitivity (82.46%), while Random Forest achieved the highest specificity (82.75%) and ROC-AUC (87.54%). Five-fold cross-validation further indicated stable performance across folds. Overall, the results demonstrate that conventional machine-learning methods, particularly SVM and Random Forest, can provide effective and computationally efficient baselines for concrete crack classification. The findings also highlight the importance of dimensionality reduction and multi-metric evaluation. The proposed framework provides a reproducible baseline for comparison with more computationally intensive deep-learning approaches.

VII. FUTURE WORK

Future work will focus on improving the robustness and generalization of conventional machine-learning-based concrete crack classification by integrating complementary handcrafted texture features such as GLCM, LBP, and HOG with pixel-based representations. Alternative feature-selection and dimensionality-reduction techniques can also be investigated to identify more discriminative representations. Further studies may evaluate ensemble and hybrid classifiers and compare their performance with transfer-learning-based deep-learning models under a consistent experimental protocol. Cross-dataset validation using additional concrete crack datasets will be important to assess generalization under different surface conditions and imaging environments. Finally, explainable-AI techniques and lightweight deployment strategies can be explored to improve model interpretability and support practical automated infrastructure inspection.

REFERENCES

- [1] Z. Chen and T. C. Hutchinson, "Image-Based Framework for Concrete Surface Crack Monitoring and Quantification," *Advances in Civil Engineering*, 2010, Art. no. 215295, doi: 10.1155/2010/215295.
- [2] T. Nishikawa, J. Yoshida, T. Sugiyama, and Y. Fujino, "Concrete Crack Detection by Multiple Sequential Image Filtering," *Computer-Aided Civil and Infrastructure Engineering*, vol. 27, no. 1, pp. 29–47, 2012, doi: 10.1111/j.1467-8667.2011.00716.x.
- [3] I. Abdel-Qader, O. Abudayyeh, and M. E. Kelly, "Analysis of Edge-Detection Techniques for Crack Identification in Bridges," *Journal of Computing in Civil Engineering*, vol. 17, no. 4, pp. 255–263, 2003, doi: 10.1061/(ASCE)0887-3801(2003)17:4(255).
- [4] A. Mohan and S. Poobal, "Crack Detection Using Image Processing: A Critical Review and Analysis," *Alexandria Engineering Journal*, vol. 57, no. 2, pp. 787–798, 2018, doi: 10.1016/j.aej.2017.01.020.
- [5] B. Wang, Q. Zhang, and W. Zhao, "Fast Concrete Crack Detection Method via L2 Sparse Representation," *Electronics Letters*, vol. 54, no. 12, 2018, doi: 10.1049/el.2018.0412.



- [6] P.-J. Chun, S. Izumi, and T. Yamane, "Automatic Detection Method of Cracks from Concrete Surface Imagery Using Two-Step Light Gradient Boosting Machine," *Computer-Aided Civil and Infrastructure Engineering*, vol. 36, no. 1, pp. 61–72, 2021, doi: 10.1111/mice.12564.
- [7] R. Wang, R.-Q. Chen, X.-X. Guo, J.-X. Liu, and H.-Y. Yu, "Automatic Recognition System for Concrete Cracks with Support Vector Machine Based on Crack Features," *Scientific Reports*, vol. 14, Art. no. 20057, 2024, doi: 10.1038/s41598-024-71075-1.
- [8] Y.-J. Cha, W. Choi, and O. Büyüköztürk, "Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, pp. 361–378, 2017, doi: 10.1111/mice.12263.
- [9] B. Kim and S. Cho, "Automated Vision-Based Detection of Cracks on Concrete Surfaces Using a Deep Learning Technique," *Sensors*, vol. 18, no. 10, Art. no. 3452, 2018, doi: 10.3390/s18103452.
- [10] 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, doi: 10.1016/j.conbuildmat.2018.08.011.
- [11] S. Dorafshan, R. J. Thomas, and M. Maguire, "SDNET2018: An Annotated Image Dataset for Non-Contact Concrete Crack Detection Using Deep Convolutional Neural Networks," *Data in Brief*, vol. 21, pp. 1664–1668, 2018, doi: 10.1016/j.dib.2018.11.015.
- [12] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang, "DeepCrack: Learning Hierarchical Convolutional Features for Crack Detection," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1498–1512, 2019, doi: 10.1109/TIP.2018.2878966.
- [13] J. Lee, H.-S. Kim, N. Kim, E.-M. Ryu, and J.-W. Kang, "Learning to Detect Cracks on Damaged Concrete Surfaces Using Two-Branched Convolutional Neural Network," *Sensors*, vol. 19, no. 21, Art. no. 4796, 2019, doi: 10.3390/s19214796.
- [14] J. Zhang, C. Lu, J. Wang, L. Wang, and X.-G. Yue, "Concrete Cracks Detection Based on FCN with Dilated Convolution," *Applied Sciences*, vol. 9, no. 13, Art. no. 2686, 2019, doi: 10.3390/app9132686.
- [15] H. Xu, X. Su, Y. Wang, H. Cai, K. Cui, and X. Chen, "Automatic Bridge Crack Detection Using a Convolutional Neural Network," *Applied Sciences*, vol. 9, no. 14, Art. no. 2867, 2019, doi: 10.3390/app9142867.
- [16] S. Li and X. Zhao, "Image-Based Concrete Crack Detection Using Convolutional Neural Network and Exhaustive Search Technique," *Advances in Civil Engineering*, 2019, Art. no. 6520620, doi: 10.1155/2019/6520620.
- [17] A. Kundu Mondal and A. Chamatkar, "Deep Learning Methodologies for Non-Destructive Evaluation of Cultural Heritage Structures," *International Journal of Mathematics and Computer Research*, vol. 14, no. 3, pp. 6199–6204, Mar. 2026, doi: 10.47191/ijmcr/v14i3.02.
- [18] T. Yamane and P.-J. Chun, "Crack Detection from a Concrete Surface Image Based on Semantic Segmentation Using Deep Learning," *Journal of Advanced Concrete Technology*, vol. 18, no. 9, pp. 493–504, 2020, doi: 10.3151/jact.18.493.
- [19] C. Su and W. Wang, "Concrete Cracks Detection Using Convolutional Neural Network Based on Transfer Learning," *Mathematical Problems in Engineering*, 2020, Art. no. 7240129, doi: 10.1155/2020/7240129.
- [20] "Automated Crack Detection and Measurement Based on Digital Image Correlation," *Construction and Building Materials*, vol. 256, Art. no. 119383, 2020, doi: 10.1016/j.conbuildmat.2020.119383.
- [21] "Concrete Crack Detection with Handwriting Script Interferences Using Faster Region-Based Convolutional Neural Network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 35, no. 4, pp. 373–388, 2020, doi: 10.1111/mice.12497.
- [22] Y. Ren et al., "Image-Based Concrete Crack Detection in Tunnels Using Deep Fully Convolutional Networks," *Construction and Building Materials*, vol. 234, Art. no. 117367, 2020, doi: 10.1016/j.conbuildmat.2019.117367.



- [23] T.-T. Le and V.-H. Nguyen, "Development of Deep Learning Model for the Recognition of Cracks on Concrete Surfaces," *Applied Computational Intelligence and Soft Computing*, 2021, Art. no. 8858545, doi: 10.1155/2021/8858545.
- [24] A. Arbaoui, A. Ouahabi, S. Jacques, and M. Hamiane, "Concrete Cracks Detection and Monitoring Using Deep Learning-Based Multiresolution Analysis," *Electronics*, vol. 10, no. 15, Art. no. 1772, 2021, doi: 10.3390/electronics10151772.
- [25] X. Meng, "Concrete Crack Detection Algorithm Based on Deep Residual Neural Networks," *Scientific Programming*, 2021, Art. no. 3137083, doi: 10.1155/2021/3137083.
- [26] L. Yu et al., "Intelligent Crack Detection and Quantification in the Concrete Bridge: A Deep Learning-Assisted Image Processing Approach," *Advances in Civil Engineering*, 2022, Art. no. 1813821, doi: 10.1155/2022/1813821.
- [27] J. Kun, Z. Zhenhai, J. Yu, and J. Dang, "A Deep Learning-Based Method for Pixel-Level Crack Detection on Concrete Bridges," *IET Image Processing*, vol. 16, pp. 2609–2622, 2022, doi: 10.1049/ipr2.12512.
- [28] A. M. Parrany and M. Mirzaei, "A New Image Processing Strategy for Surface Crack Identification in Building Structures Under Non-Uniform Illumination," *IET Image Processing*, vol. 16, no. 9, 2022, doi: 10.1049/ipr2.12357.
- [29] P. Palevičius et al., "Automatic Detection of Cracks on Concrete Surfaces in the Presence of Shadows," *Sensors*, vol. 22, no. 10, Art. no. 3662, 2022, doi: 10.3390/s22103662.
- [30] E. A. Shamsabadi et al., "Vision Transformer-Based Autonomous Crack Detection on Asphalt and Concrete Surfaces," *Automation in Construction*, vol. 140, Art. no. 104316, 2022, doi: 10.1016/j.autcon.2022.104316.
- [31] P. Shokri, M. Shahbazi, and J. Nielsen, "Semantic Segmentation and 3D Reconstruction of Concrete Cracks," *Remote Sensing*, vol. 14, no. 22, Art. no. 5793, 2022, doi: 10.3390/rs14225793.
- [32] R. Kirthiga and S. Elavenil, "A Survey on Crack Detection in Concrete Surface Using Image Processing and Machine Learning," *Journal of Building Pathology and Rehabilitation*, vol. 9, no. 1, 2024, doi: 10.1007/s41024-023-00371-6.
- [33] İ. Akgül, "Mobile-DenseNet: Detection of Building Concrete Surface Cracks Using a New Fusion Technique Based on Deep Learning," *Heliyon*, vol. 9, no. 10, Art. no. e21097, 2023, doi: 10.1016/j.heliyon.2023.e21097.

