SSRG International Journal of Civil Engineering
ISSN: 2348-8352/ https://doi.org/10.14445/23488352/IJCE-V13I5P117

Volume 13 Issue 5, 260-267, May 2026

Original Article

A Machine Learning Model Based on MobileNet V2 and IGOA-XGBoost for Structural Health Monitoring

Ida Barkiah[1], Nurul Fathanah Mustamin[2], Yuslena Sari[3], Ayuddin[4], Amry Dasar[5], Muhammad Tommy Maulidyanto[6]

[1]*Department of Civil Engineering, Universitas Lambung Mangkurat, South Kalimantan, Indonesia.*
[2,3]*Department of Information Technology, Universitas Lambung Mangkurat, South Kalimantan, Indonesia.*
[4]*Department of Civil Engineering, Faculty of Engineering, Universitas Negeri Makassar, Makassar, Indonesia.*
[5]*Department of Civil Engineering, Universitas Sulawesi Barat, West Sulawesi, Indonesia.*
[6] *Department of Mining Engineering, Politeknik Negeri Banjarmasin, South Kalimantan, Indonesia.*

[1]*Corresponding Author : idabarkiah@ulm.ac.id*

Received: 17 March 2025 Revised: 05 May 2025 Accepted: 22 April 2026 Published: 29 May 2026

Abstract *- Monitoring the structural health of infrastructure such as buildings, bridges, and roads requires efficient and accurate crack detection methods. This research proposes the integration of MobileNet V2 for concrete crack image feature extraction with the Improved Grasshopper Optimization Algorithm (IGOA) and XGBoost for classification. MobileNet V2 was chosen for its efficiency in feature extraction, while IGOA was implemented to optimize the hyperparameters of XGBoost, improving classification performance. The dataset used includes 40,000 concrete crack images with a proportion of 70% training data and 30% test data. The experimental results show that the IGOA-XGBoost method provides the best performance with 99.75% accuracy, 99.80% precision, 99.70% recall, and 99.75% F1-score. The advantage of this method lies in IGOA's adaptive metaheuristic optimization mechanism, which balances exploration and exploitation in the search for optimal hyperparameters. This research proves the potential integration of MobileNet V2 and IGOA-XGBoost as an effective solution for an accurate and efficient machine learning-based crack detection system, supporting the development of real-world structural health monitoring systems.*

Keywords *- Concrete Crack Classification, Feature Extraction, Hyperparameter Optimization, IGOA-XGBoost, MobileNet V2.*

1. Introduction

Structural health monitoring is essential to ensure the safety and longevity of critical infrastructure, especially buildings, bridges, and roads. Concrete cracks are a common factor of structural damage, and their early detection is important to prevent catastrophic failures and costly repairs. Traditional manual inspection methods, although widely used, are often time-consuming, subjective, prone to human error, and inefficient for large-scale infrastructure. Automated crack detection systems using machine learning and computer vision have been combined as chosen alternatives, capable of analyzing large volumes of concrete surface images with higher accuracy and speed.

Despite advances in deep learning techniques for crack detection, several challenges remain unaddressed. Images of concrete cracks often display complex and varied patterns affected by environmental factors (lighting conditions, surface textures, and noise), which can degrade classification accuracy. Furthermore, many existing approaches rely on heavy neural network architectures that demand substantial computational resources, limiting their applicability in real-time or resource-constrained environments. There is a clear research gap in enhancing a crack detection system that balances high accuracy, computational efficiency, and robustness against real-world variations.

The analysis addresses the gaps by choosing an integrated approach that combines MobileNet V2, a lightweight and efficient convolutional neural network architecture, with the Improved Grasshopper Optimization Algorithm (IGOA) for hyperparameter tuning of the XGBoost classifier. MobileNet V2 is selected for its ability to extract robust features with minimal computational overhead, making it appropriate for deployment on devices with limited resources. The IGOA optimizes the XGBoost hyperparameters, enhancing classification performance and model stability. This integrated method is made to enhance the concrete crack classification efficiency and accuracy, overcoming the limitations of existing models and contributing to scalable and reliable structural health monitoring solutions.

Structural health monitoring is important to ensure the safety and longevity of infrastructure, especially buildings, roads, and bridges. The use of reliable detection for automatic detection applications is highly desirable. MobileNetV2 is an algorithm used for digital image feature extraction that is known to be reliable in increasing accuracy. [1, 2] . However, reliable accuracy is not the only solution required. Resource-efficient algorithms are also a research challenge for crack detection. The IGOA is known as an optimization algorithm that improves the performance of XGBoost in terms of resource efficiency.

Several studies have applied detailed learning architectures for crack recognition, such as the traditional convolutional neural networks and the application of transfer learning models. For instance, Yang (2024) utilized MobileNet V2 with transfer learning to improve crack detection accuracy, while Shahin et al. (2024) explored hybrid visual transformer models for enhanced feature representation. However, these approaches often lack optimization in classifier hyperparameters or require substantial computational resources. Moreover, optimization techniques used in previous studies, such as standard genetic algorithms or grid search, tend to be time-consuming and may fall into local optima.

In contrast, this study introduces a new way of combining MobileNet V2 for efficient feature extraction with the IGOA to tune the hyperparameters of the XGBoost classifier.

As far as we know, using IGOA to adjust XGBoost for concrete crack classification along with MobileNet V2 for feature extraction hasn`t been done before, which makes this a big step forward in the field.

2. Literature Review

Concrete crack detection has been widely studied using different image processing and machine learning methods. Early methods used manual feature extraction along with traditional classifiers, but they had problems because of the difficulty in creating good features. Recently, deep learning models like CNNs have shown better results because they can automatically learn useful features from images. Some studies, such as Yang (2024) and Shahin et al. (2024), have used MobileNet V2 and transformer-based models for crack classification, and they have had good success. Other research has looked into using ensemble learning methods like XGBoost for classification, often with basic optimization techniques like grid search or genetic algorithms. However, these models can sometimes be slow or not perform well because of poor hyperparameter settings, which can make them less useful in real-world situations.

Metaheuristic optimization algorithms have been applied to improve hyperparameter selection in machine learning models, yet the use of improved algorithms like IGOA remains limited in structural health monitoring contexts. This highlights the need for further research combining efficient feature extraction, robust classification, and advanced optimization methods to improve crack detection accuracy and efficiency.

State-of-the-Art Methods for Concrete Crack Classification Deep Learning Models ReCRNet: This model outperforms AlexNet, VGG19, SVM, and decision tree in terms of precision, accuracy, F1-score, recall, and ROC, making it effective for historical buildings' automatic monitoring [3]. Res-FPN-SqueezeNet: Integrates features from ResNets and FPNs for multiscale feature extraction, achieving high precision and geometric accuracy, ideal for real-time applications [4].

Transfer Learning Models TL_GoogleNet: Achieves a crack detection accuracy of up to 99. VGG16 and MobileNet: Pretrained CNNs used as feature extractors combined with regression algorithms like SVR and XGBoost for recognizing crack geometric features, showing high accuracy in diverse images [5]. VGG16, ResNet50, and MobileNetV2 with ensemble strategies like weighted average, stacking, Adaboost, and XGBoost significantly improves precision and accuracy [6]. Cascade Broad Neural Network: This architecture outperforms mainstream classification techniques in testing accuracy and training time, making it efficient for automated classification [7].

Comparison with MobileNet V2 and IGOA-XGBoost MobileNet V2: Known for its lightweight architecture and efficiency, MobileNet V2 is effective for mobile platforms' real-time applications with limited computational resources [8, 9]. The proposed MobileNet V2 and IGOA-XGBoost model is competitive with existing techniques, particularly in precision, accuracy, and suitability for real-time applications. Then, because of the lightweight architecture and efficient performance, it becomes a strong candidate for practical deployment in concrete crack classification tasks.

3. Methods

3.1. Research Flow

Research Flow starts from 1) data collection, 2) preprocessing, 3) feature extraction, 4) classification with optimization, and 5) evaluation stage. The research flowchart is presented in Figure 1.

3.2. Data Sets Collection

The image dataset applied consists of images of cracks on concrete surfaces, which were obtained from publicly available sources or collected directly in a controlled environment. Meanwhile, to ensure data diversity and improve the accuracy of the model in recognizing crack patterns, some images were also taken in a controlled environment using a high-resolution camera with optimally controlled lighting settings.

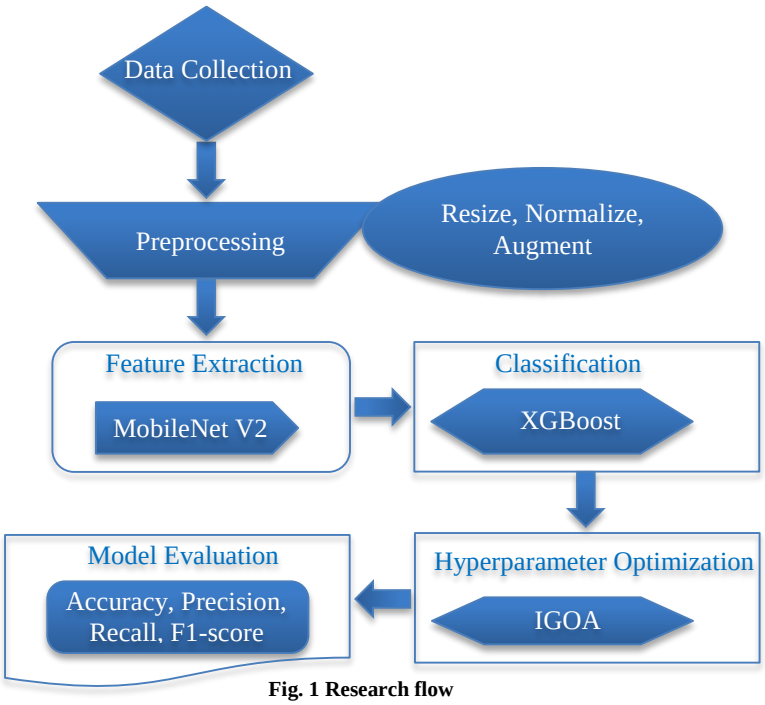


Fig. 1 Research flow

The dataset consists of 40,000 images with a 70:30 split. The target data consists of two types: negative and positive. The image size is 227x277 pixels with a color range of 256*256*256 = 16777216 possible colors. The color types are Red, Green, and Blue.

3.3. Pre-Processing

The image is cropped to a size of 244x244 during pre-processing. This step is necessary to support the efficiency of the feature extraction process. [10] . By maintaining uniform dimensions, the model can recognize patterns and features in the images more consistently and avoid the problem of scale differences that can impact the results of classification accuracy. The training and testing data division is 70:30. Future work may include explicit validation sets or k-fold cross-validation to further enhance the generalization assessment.

In addition, image size normalization also helps in improving computational efficiency during the model training process [11]. If the image size is allowed to vary, then the model will require an additional process to adjust the dimensions of each image before going into the feature extraction stage, which can extend the computation time and increase the processing power requirement [12].

Therefore, each image is converted to a standard resolution before passing the feature extraction stage using MobileNet V2, so that the classification process can run more optimally and produce more accurate predictions. The following are the pre-processing steps:

- Normalization: The pixel values in each image are normalized into the range [0,1] by grouping them by the maximum value of 255 [13]. This normalization aims to improve the efficiency of model training by ensuring a more uniform distribution of data, thus preventing explosive gradients or vanishing gradients in artificial neural networks. With smaller data scales, optimization algorithms such as Adam or SGD can work more efficiently, speed up model convergence, and improve accuracy without requiring excessive iterations [14]
- Data Augmentation: The augmentation stage provides a variety of image angles. This stage is carried out with the aim of increasing accuracy [15, 16]. With this augmentation, the model becomes more robust, able to recognize patterns under various conditions, and reduces the risk of overfitting, thus improving the accuracy in detecting features in new data.

3.4. Feature Extraction Using MobileNet V2

MobileNet V2 is applied as the feature extraction method in this study due to its lightweight, efficient design, and it is optimized for devices with computational limitations, such as smartphones and edge computing-based systems. [17, 18] . MobileNetV2 excels in accuracy for feature extraction layers in Convolutional Neural Networks (CNNs). It is designed with balanced weights, resulting in reliable accuracy.

MobileNetV2 works with a parallel version for feature extraction using distillation knowledge and can optimize performance on small datasets. This efficient algorithm works optimally, although efficiency can reduce the accuracy performance. [12].

3.5. Classification Using IGOA-XGBoost

XGBoost is a classification algorithm derived from GBDT (Gradient Boosting Decision Trees). This algorithm is known for its speed in image classification.[19] XGBoost is a regression algorithm that excels in efficiency and consistently provides the most accurate results.[20] . With this technique, XGBoost is superior to other boosting algorithms, such as AdaBoost or traditional Gradient Boosting. Another advantage of XGBoost is its ability to handle data with a large number of features, including data containing missing values. XGBoost can automatically handle incomplete data by performing optimal path selection in the decision tree[21].

The IGOA is a meta-heuristic optimization technique following the grasshopper swarm's behavior. It is particularly useful for handling complex optimization problems, especially hyperparameter tuning for machine learning models like XGBoost. Here is a step-by-step guide on how to implement GOA with XGBoost in Python, along with theoretical foundations, practical coding steps, performance comparisons, and potential applications.

- Theoretical Foundations of IGOA. (1) Inspiration: IGOA mimics the swarming behavior of grasshoppers, which allows it to explore and exploit the search space effectively. However, the original GOA can struggle with local optima and slow convergence rates due to its linear convergence parameter. (2) Improvements: Enhanced versions of GOA, such as the Improved Grasshopper Optimization Algorithm (IGOA) and Adaptive Average GOA (AAGOA), introduce mechanisms like nonlinear convergence parameters and random jumping strategies to improve exploration and convergence speed.
- Steps to Implement GOA with XGBoost in Python. (1) Data Preprocessing: Feature Selection: Apply methods like correlation analysis/recursive feature elimination to choose relevant features for your model. (2) Implement the GOA algorithm to search for optimal hyperparameters. Evaluating the fitness of each grasshopper based on the XGBoost model performance. (3) Training the Model: o Use the optimal hyperparameters found by GOA to try the XGBoost model. (4) Best Practices: (1) Cross-Validation: Use k-fold cross-validation to make sure that the model generalizes well to unseen data. (2) Parameter Constraints: Set reasonable bounds for hyperparameters to avoid unrealistic values during optimization.

The XGBoost‘s corresponding outputs

$$\hat{y}\ i(0) = 0 \tag{1}$$

$$\hat{y}\ i(1) = f1(xi) = \hat{y}_i^{(0)} + f1(xi) \tag{2}$$

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} = \hat{y}_i^{(t-1)} + f1(xi) \tag{3}$$

$$\hat{y}\ i = \sum_{k=1}^{k} f_x(x_i), (i = 1,2, \dots, n) + \tag{4}$$

In equation (1d), $f\ (x)$ is $f\ (x) = w\ q(x)$ ($q : \mathbb{R}m \rightarrow 1, 2, ..., T$, $w \in \mathbb{R}T$) where q is the tree structure with leaf nodes. T is the total number of leaf nodes, and w is the leaf weights.

$$\Omega^t = \sum_{i=1}^{n} 1\left(y_i, \hat{y}_i^{t-1} + f_t(x_i)\right) + yT + 1/2\,\lambda \sum_{j=1}^{T} w_j^2 \tag{5}$$

$$l = yilog(\sigma(\hat{y}\ i(t-1) + ft(xi))) + (1-yi)log(1-\sigma(\hat{y}\ i(t-1) + ft(xi))) \tag{6}$$

$$\sigma(z)=1/(1+e-z) \tag{7}$$

The objective function can be seen in Equation(2a), where the first term shows the loss function, which is the log-likelihood of the Bernoulli distribution. The second term, γT, states the L1 regularization term, and γ states the minimum split loss reduction [5]. The last is the L2 regularization term. When the model optimizes the training data, the objective function is minimized by giving each tree a learning rate. Since the model is iterative, each tree is multiplied by the learning rate and given a value at each iteration.

In addition, the optimization method used in XGBoost, which is based on parallel processing, makes it faster than other boosting models. Training was done using the initial boosting method for 20 rounds without any increase in classification errors on the validation dataset, to prevent overfitting and ensure the model performs at its best.

3.6. Hyperparameter Optimization with IGOA

The Improved Grasshopper Optimization Algorithm (IGOA) measures the potential solutions' effectiveness by evaluating the performance against problem-specific criteria, guiding the algorithm in finding the proper solution. IGOA is applied to challenging optimization problems. The algorithm mathematically models and mimics the grasshopper swarm's behavior to solve the optimization challenge. Another key difference between IGOA and other optimization algorithms is that it improves the particle positions relative to the current position as well as personal and global best. Although GOA updates the search agent's position based on the current position, global best, and all other search agents' positions. Therefore, in other optimization algorithms, no other particles contribute to updating the particle's position, whereas GOA needs all agents to obtain each agent’s next position.[22, 23] The algorithm was designed to mathematically copy how grasshoppers move together in nature to solve complex problems. It created a model that shows how grasshoppers push away from each other and pull towards one another. The pushing helps them look for new areas to search, while the pulling helps them focus on good areas they find. To make sure they don't get stuck in one place, the algorithm uses a changing number that makes the grasshoppers' comfort zone smaller over time. In the end, the best solution found by the group is treated as a goal that the grasshoppers keep trying to improve.

The main advantage of IGOA over the standard GOA lies in the search strategy improvement mechanism, which includes refinements in the control of the exploration speed parameter, enhancement of the adaptation mechanism of the movement of individuals in the population, as well as the implementation of a more dynamic movement strategy to accelerate convergence to the optimal solution.[24] . In other words, IGOA is not only able to explore the hyperparameter space more effectively but also accelerates the process of finding the optimal solution, reduces computation time, and improves the performance of XGBoost in handling complex data. In this study, IGOA helps determine the optimal hyperparameter combination so that XGBoost can work with higher efficiency in the classification process.

The Grasshopper Optimization Algorithm plays a crucial role in optimizing hyperparameters for XGBoost models by providing a robust and adaptable global search mechanism. Its effectiveness is further enhanced when combined with other optimization techniques, leading to significant improvements in model performance and convergence rates. This makes GOA a precious tool for hyperparameter tuning in complex machine learning tasks.

3.7. Performance Evaluation

The dataset was grouped into 70% training and 30% testing sets randomly, resulting in 28,000 training and 12,000 testing images. Model performance was assessed by applying standard classification metrics (accuracy, precision, recall, and F1-score), which were calculated from the confusion matrix generated by the Scikit-learn library in Python. The formulas used are:

- Accuracy=(TP + TN)/(TP + TN + FP + FN)
- Precision=TP/(TP + FP)
- Recall=TP/(TP + FN)
- F1-Score =2×(Precision×Recall)/(Precision+Recall)
  where TP=true positive, TN=true negative, FP= false positive, and FN=false negative counts.

4. Results and Discussion

4.1. Code

MobileNet v2 uses the TensorFlow library, and XGBoost in Python was selected because it is commonly applied in machine learning and offers open-source libraries on GitHub. The program begins by using Pillow for image preprocessing, which involves resizing the image to 224x224 pixels and turning it into an array. The image array is batch dimensioned and processed with preprocess_input according to the deep learning model used, MobileNet v2. The MobileNet v2 model is retrieved through the MobileNetV2 library available in TensorFlow. The feature extraction results are stored in Pandas DataFrame format using pd.DataFrame() and saved as CSV. With the Pandas DataFrame format, the extraction results from MobileNet v2 can be divided into train and test data by applying train_test_split. Furthermore, XGBoost is used as a classifier for training. In the testing stage, the metric functions of Scikit-learn, namely precision_score, f1_score, accuracy_score, and recall_score, are used to provide an assessment of the classifier's performance.

4.2. Research Tools

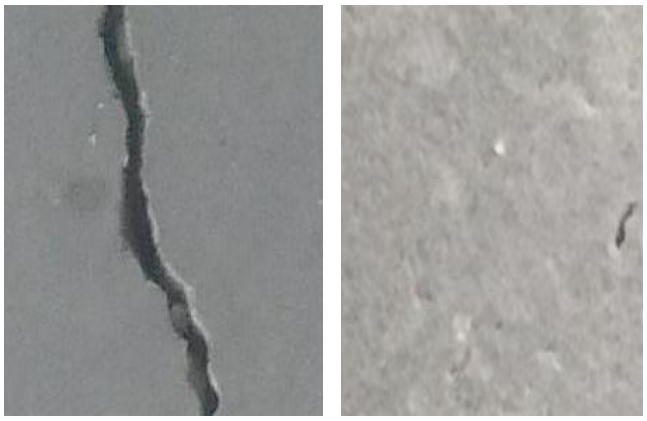
This research uses hardware with a detailed explanation in Table 1: Hardware Specifications.

Table 1. Hardware Specification Table

Laptop
11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz (8 CPUs), ~2.8GHz Intel(R) Iris(R) Xe Graphics
16384MB RAM
Python 3.9.19

4.3. Image Acquisition

The data collected is 40,000 image data measuring 227x227 pixels with RGB channels, each of which consists of 20,000 data points for each class, namely 20,000 data points for images with positive labels and 20,000 for images with negative labels. The data is obtained from Kaggle.[25]



(a) **(b)**

Fig. 2 (a) Positive, and (b) negative concrete images.

The 40,000 data will be divided randomly between training and test, where 70% for training and 30% for test, resulting in 28,000 training and 12,000 test data. From each data set, performance evaluation will be conducted by applying a confusion matrix to obtain the recall, F1-score, precision, and accuracy value.

4.4. Feature Extraction

The image feature extraction stage is performed using a lightweight model of *deep learning*, MobileNetV2, which is used to identify and extract important features from images. MobileNet v2 does so using an efficient and compact architecture, which relies on a *depthwise separable convolutions* technique. This technique splits the convolution process into *depthwise, which* processes each channel

separately, and a pointwise that mixes the channels. In addition, MobileNet v2 implements *inverted residuals*, a block that allows for better information exchange between layers with *shortcut connections*, therefore developing the model's effectiveness and performance. The *library* used is Tensorflow (tf.keras.applications.MobileNetV2 | TensorFlow v2.16.1).

The data from feature extraction results in 1,280 features, so the data becomes 40,000 *rows* x 1,280 *columns*. Some of the results are shown in Table 2.

Table 2. HOG Feature Extraction Table

0	1	2	...	1277	1278	1279	Label
0.241068	0.000000	0.041763	...	0.0	0.780682	0.391599	Positive
0.985650	0.042202	0.003978	...	0.0	0.408269	0.230168	Positive
0.147621	0.000000	0.017295	...	0.0	0.209309	0.140560	Positive
0.421567	0.000000	0.040216	...	0.0	0.275184	0.183305	Positive

4.5. Classification

The classification stage is conducted by applying the *ensemble* technique, XGBoost, which is a machine learning algorithm under the Gradient Boosting framework. The application of XGBoost in the Python programming language applies a *library* with the same name, xgboost. The implementation can be seen below:

```
import xgboost as xgb
xgb_classifier = xgb.XGBClassifier()
```

4.6. IGOA-XGBoost

The Improved Gazelle Optimization Algorithm (IGOA) used above is a metaheuristic algorithm that follows the hunting behavior of a gazelle (deer) more efficiently to obtain the proper solution. First, an initial population of a certain size is randomly initialized within the constraints of predefined parameters. Each individual in this population represents a set of hyperparameters for the XGBoost model. At each iteration, all individuals in the population are judged based on the fitness value calculated using the accuracy of the XGBoost model trained with those hyperparameters. The best individual (highest accuracy) is kept as the best temporary solution. Thereafter, the population is updated using two random vectors, which are used to update the individual's position to explore and exploit better solutions. This process is repeated until it reaches a predetermined total number of iterations. Finally, the best solution found during the iterations is considered the optimal set of hyperparameters. In each iteration, the best performance (based on accuracy) as well as the associated hyperparameters are recorded to ensure transparency and analysis of the solution search progress. The algorithm is adapted to maximize the accuracy of the XGBoost model on binary classification tasks by considering parameters like the total estimators, learning rate, max depth, and subsampling.

4.7. Method Performance Testing

This analysis tests the feature extraction and classification accuracy that has been done using a *confusion matrix*. The method performance evaluation is done by applying the *default* parameters obtained through the XGBoost *library*. Testing is done using the scikit-learn *library* found in the Python programming language. From the *library*, the *precision*, *accuracy*, *F1 score*, and *recall* values are sought.

Accuracy is the overall data's right prediction value in the *confusion matrix*. *Precision* is the positive predictive value of all positive predicted results. *Recall* is the speed and computational efficiency value of right positive predictions compared to all data. Meanwhile, the *F1* score is a weighted average comparison of *precision* and *recall*.

The program code written in Python is:

```
accuracy_score (label, pred)
f1_score(label, pred, average =  macro)
precision_score(label, pred, average =  macro)
recall_score(label, pred, average =  macro)
```

The results of the *precision*, *recall*, *f1-score*, and *accuracy* values can be seen in the MobileNetV2 and IGOA-XG Boost performance evaluation results table.

Table 3. Results Table of MobileNet V2 and IGOA-XGBoost

Accuracy	F1 Score	Precision	Recall
0.99750	0.99750	0.99799	0.99700

XGBoost is compared with other *ensemble* methods with parameters that have been determined by *default*. The results of the method comparison can be seen in the XGBoost performance comparison results table.

Table 4. XGBoost performance comparison

Methods	Accuracy	F1 Score	Precision	Recall
XG Boost	0.99566	0.99533	0.99600	0.99567
Random Forest	0.99475	0.99384	0.99566	0.99475
AdaBoost	0.99316	0.99315	0.99564	0.99067
IGOA-XG Boost	**0.99750**	**0.99750**	**0.99799**	**0.99700**
Gradient Boosting	0.99250	0.99250	0.99299	0.99200
Histogram Gradient Boosting	0.99383	0.99383	0.99350	0.99416

The best precision, recall, f1-score, and accuracy values are obtained in the IGOA-XGBoost method. IGOA-XGBoost has an advantage over regular XGBoost and other ensemble methods because of its approach to optimizing hyperparameters. IGOA uses a metaheuristic mechanism that

mimics gazelle behavior to search for better solutions in the hyperparameter search space adaptively. IGOA dynamically balances between exploration (new solutions) and exploitation (improving good solutions), thus finding more optimal hyperparameter sets faster. This helps enhance the XGBoost model performance. MobileNet v2 has been shown to work well in combination with ensemble methods, especially in image classification tasks.

Boosting methods allow models to utilize predictions from multiple networks, thus improving accuracy and reducing overfitting. The use of MobileNet v2 as feature extraction in an ensemble has resulted in improved classification accuracy on datasets such as ImageNet and CIFAR-10. This shows that although MobileNet v2 is known for its efficiency, the use of an ensemble can further improve its performance without sacrificing much of that efficiency [26]. The manual measurement for the IGOA-XGBoost confusion matrix from MobileNet v2 feature extraction can be seen as follows:

Table 5. Confusion Matrix Table

	Positive	**Negative**
Positive	5985	12
Negative	18	5985

Then, the calculations are:

- $Accuracy = \frac{TP+TN}{(TP+FP+TN+FN)} = \frac{5985+5985}{(5985+12+5985+18)} = 0.9975 = 99.75\%$
- $Recall = \frac{TP}{(TP+FN)} = \frac{5985}{(5985+18)} = \frac{5985}{6003} = 0.997001499 = 99.70\%$
- $Precision = \frac{TP}{(TP+FP)} = \frac{5985}{(5985+12)} = \frac{5985}{5997} = 0.997998999 = 99.80\%$
- $F1-score = \frac{2\,(Recall\ x\ Precision)}{(Recall+Precision)} = \frac{2\,(0.997001499\ x\ 0.997998999)}{(0.997001499+0.997998999)} = 0.997500001 = 99.75\%$

Based on the equation, the accuracy, recall, precision, and f1-score values of MobileNet v2 IGOA-XGBoost are 99.75%, 99.70%, 99.80%, and 99.75%, respectively.

With the results of the method performance obtained, it requires approximately ±6 hours for each data experiment, calculating 40,000 *rows* x 1,280 *columns*, conducted to apply MobileNet v2 as a feature extraction and IGOA-XGBoost as a classification technique (*machine learning)* with optimized *hyperparameters.*

The proposed integration of MobileNet V2 for feature extraction with IGOA-XGBoost for classification demonstrates superior performance compared to several conventional ensemble techniques (Random Forest, AdaBoost, and standard XGBoost), as shown in Table 4. The method provides 99.75% accurac, which is 0.18% higher than the best method. The adaptive IGOA hyperparameters can adapt to achieve high accuracy. This leads to better accuracy and faster computation.

5. Conclusion

Based on the analysis of the document titled "Crack Image Classification with MobileNet V2 Feature Extraction and IGOA-XGBoost," this research presents a method that is both efficient and accurate for identifying and grouping cracks on concrete surfaces. The study uses MobileNet V2, which is a known model that works well because it is lightweight and fast at extracting features, along with the Improved Grasshopper Optimization Algorithm (IGOA) and XGBoost for classification. The use of IGOA helps improve the model's settings, making it more dependable and able to classify crack images more accurately, even when lighting or surface types change. While the study mainly looks at cracks in concrete, the mix of MobileNet V2 with IGOA and XGBoost might also be helpful for other tasks related to checking the condition of structures, such as finding problems like rust, surface damage, or bending in different parts of buildings and other infrastructure. Even though this study shows that MobileNet V2 works well with IGOA and XGBoost for crack detection, there are still several areas that could be explored further in future research.

Conflicts of Interest

The authors declare no conflicts of interest.

Funding Statement

This research was funded by Universitas Lambung Mangkurat Research Grant, grant number 6.1330/UN8.2/PG/2024

Acknowledgments

The authors gratefully acknowledge the use of the Laboratorium Big Data at Universitas Lambung Mangkurat.

References

[1] Mohammad Shahin et al., "Improving the Concrete Crack Detection Process via a Hybrid Visual Transformer Algorithm," *Sensors*, vol. 24, no. 10, pp. 1-32, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[2] Fujie Yang, "Enhancing Concrete Crack Image Detection Using MobileNetV2 and Transfer Learning," *Frontiers in Science and Engineering*, vol. 4, no. 4, pp. 110-119, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[3] Hatice Catal Reis, and Kourosh Khoshelham, "ReCRNet: A Deep Residual Network for Crack Detection in Historical Buildings," *Arabian Journal of Geosciences*, vol. 14, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] Chunzheng Zhang et al., "Res-FPN–SqueezeNet Segmentation Architecture for Enhanced Identification of Concrete Cracks within a Deep-Learning Framework," *Journal of Computing in Civil Engineering*, vol. 39, no. 5, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[5] Miao Su et al., "Utilizing Pretrained Convolutional Neural Networks for Crack Detection and Geometric Feature Recognition in Concrete Surface Images," *Journal of Building Engineering*, vol. 98, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[6] Ali Mayya et al., "Efficient Hybrid Ensembles of CNNs and Transfer Learning Models for Bridge Deck Image-Based Crack Detection," *Structures*, vol. 64, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[7] Li Guo, Runze Li, and Bin Jiang, "A Cascade Broad Neural Network for Concrete Structural Crack Damage Automated Classification," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2737-2742, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Ganesh Kolappan Geetha, and Sung-Han Sim, "Fast Identification of Concrete Cracks using 1D Deep Learning and Explainable Artificial Intelligence-Based Analysis," *Automation in Construction*, vol. 143, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[9] Lijie Zhao et al., "Transfer Learning and Multi-Level Image Processing for Crack Classification and Quantitative Measurement in Concrete Structures Under Complex Contexts," *Structures*, vol. 80, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[10] MobileNet, MobileNetV2, and MobileNetV3, Keras 3 API Documentation, 2024. [Online]. Available: https://keras.io/api/applications/mobilenet/

[11] Nuwan Madusanka et al., "Impact of H&E Stain Normalization on Deep Learning Models in Cancer Image Classification: Performance, Complexity, and Trade-Offs," *Cancers*, vol. 15, no. 16, pp. 1-17, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Sergio Saponara, and Abdussalam Elhanashi, "Impact of Image Resizing on Deep Learning Detectors for Training Time and Model Performance," *Applications in Electronics Pervading Industry, Environment and Society*, vol. 866, pp. 10-17, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Andrea Cina et al., "Comparing Image Normalization Techniques in an End-to-End Model for Automated Modic Changes Classification from MRI Images," *Brain and Spine*, vol. 4, pp. 1-6, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[14] Florian June, Optimization Algorithm: From SGD to Adam, Medium, 2023. [Online]. Available: https://medium.com/@florian_algo/optimization-algorithm-from-sgd-to-adam-50ea22187951

[15] Eman M.G. Younis et al., "A Hybrid Deep Learning Model with Data Augmentation to Improve Tumor Classification Using MRI Images," *Diagnostics*, vol. 14, no. 23, pp. 1-14, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[16] Wenguan Wang, Jianbing Shen, and Haibin Ling, "A Deep Network Solution for Attention and Aesthetics Aware Photo Cropping," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 7, pp. 1531-1544, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[17] Muneeb A. Khan, Heemin Park, and Jinseok Chae, "A Lightweight Convolutional Neural Network (CNN) Architecture for Traffic Sign Recognition in Urban Road Networks," *Electronics*, vol. 12, no. 8, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Liquan Zhao et al., "A Lightweight Deep Neural Network with Higher Accuracy," *PLoS One*, vol. 17, no. 8 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] Zeravan Arif Ali et al., "Exploring the Power of eXtreme Gradient Boosting Algorithm in Machine Learning: a Review," *Academic Journal of Nawroz University (AJNU)*, vol. 12, no. 2, pp. 320-334, 2023. [Google Scholar] [Publisher Link]

[20] Shangkun Deng et al., "Stock Price Crash Warning in the Chinese Security Market Using a Machine Learning-Based Method and Financial Indicators," *Systems*, vol. 10, no. 4, pp. 1-25, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[21] Vimal Rathakrishnan, Salmia Beddu, and Ali Najah Ahmed, "Comparison Studies Between Machine Learning Optimisation Techniques on Predicting Concrete Compressive Strength," *Research Square*, pp. 1-54, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[22] Adil Mehdary et al., "Hyperparameter Optimization with Genetic Algorithms and XGBoost: A Step Forward in Smart Grid Fraud Detection," *Sensors*, vol. 24, no. 4, pp. 1-24, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[23] Peng Qin, Hongping Hu, and Zhengmin Yang, "The Improved Grasshopper Optimization Algorithm and its Applications," *Scientific Reports*, vol. 11, pp. 1-14, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[24] Wei Liu et al., "A Multi-Strategy Improved Grasshopper Optimization Algorithm for Solving Global Optimization and Engineering Problems," *International Journal of Computational Intelligence Systems*, vol. 17, pp. 1-28, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[25] Çağlar Fırat Özgenel, Concrete Crack Images for Classification, Kaggle, 2019. [Online]. Available: https://www.kaggle.com/datasets/arnavr10880/concrete-crack-images-for-classification

[26] Mark Sandler et al., MobileNetV2: Inverted Residuals and Linear Bottlenecks, [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/papers/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.pdf